Deep Neural Network Approach for Single Channel Speech Enhancement Processing

Dongfu Li

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for degree

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Abstract

Speech intelligibility represents how comprehensible a speech is. It is more important than speech quality in some applications. Single channel speech intelligibility enhancement is much more difficult than multi-channel intelligibility enhancement. It has recently been reported that training-based single channel speech intelligibility enhancement algorithms perform better than Signal to Noise Ratio (SNR) based algorithm. In this thesis, a training-based Deep Neural Network (DNN) is used to improve single channel speech intelligibility. To increase the performance of the DNN, the Multi-Resolution Cochlea Gram (MRCG) feature set is used as the input of the DNN. MATLAB objective test results show that the MRCG-DNN approach is more robust than a Gaussian Mixture Model (GMM) approach. The MRCG-DNN also works better than other DNN training algorithms. Various conditions such as different speakers, different noise conditions and reverberation were tested in the thesis.

Acknowledgements

I am thankful to all those people who helped, advised, and accompanied me during my studies and my research, who made this thesis possible and gave me an unforgettable experience that I will treasure for all my life.

I would like to express my deepest and sincerest gratitude to my supervisor Prof. Martin Bouchard, who gave me such a great opportunity of taking part in the Master of Applied Science program. His help, valuable guidance and professionalism are the real sources that kept my research going. At the same time, he gave me the freedom to explore on my own, and definitely provided me with the tools that I needed to choose the right direction and successfully complete my thesis.

I am thankful to the classmates and friends that I made during my studies. Their advice and help made the entire class time colorful, and enriched my after-school life.

Many thanks to the colleagues in the research laboratory SITE 5130, they helped me a lot during my research time as well as in my daily life.

Best wishes to my lovely wife Rui Bao, her patience and encouragement support me through all the difficulties in my life.

Finally, I would like to thank my parents for supporting me throughout all my studies at University.

Table of contents

List of figures

List of tables

Abbreviations

AMS	Amplitude Modulation Spectrogram
ANSI	American National Standards Institute
CE	Cross-Entropy
CSII	Coherence Speech Intelligibility Index
CVC	Consonant-Vowel-Consonant
DBN	Deep Belief Network
DNN	Deep Neural Network
FA	False Alarm probability
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HIT	correct detection probability
MFC	Mel-Frequency Cepstrum
MFCC	Mel-Frequency Cepstral Coefficient
MMSE	Minimum Mean Square Error
MRCG	Multi-Resolution Cochlea Gram
MSE	Mean Squared Error
MTF	Modulation Transfer Function
NN	Neural Network
PLP	Perceptual Linear Prediction
RASTA-PLP	Relative Spectral Transform-PLP,
RMS	Root-Mean-Square
SII	Speech Intelligibility Index
SNR	Signal to Noise Ratio
STI	Speech Transmission Index
STFT	Short-Time Fourier transform
T-F	Time Frequency

Chapter 1. Introduction

Speech intelligibility is a measure of how comprehensible the speech is in a given condition[1]. In the ISO 9921 standard, speech intelligibility is defined as 'a measure of effectiveness of understanding speech'. In some cases, to have a target speech understandable outweighs the quality of the speech. For example, in public areas, such as airports, railway stations, or shopping malls, a series of tragic consequences may result if the announcement of an emergency is misunderstood. Speech intelligibility plays an important role in the evaluation of hearing aids and cochlear implants, simply because the main purpose of using these devices is to let the user understand others more easily.

With an increase of background noise and reverberation, speech intelligibility decreases. Because of different frequency content, stationary noise like white noise or pink noise cause different effects for speech intelligibility compared to non-stationary noise like cocktail party babble noise or multi-talker noise. Except for simple cases where speech and noise can be easily separated in time or frequency, traditional single channel noise reduction methods can improve speech quality but not speech intelligibility, for reasons that are still not entirely understood.

Single-channel speech intelligibility enhancement is more challenging than multi-channel speech intelligibility enhancement, because information about the spatial sound propagation is not available. In most cases, it's hard for a single-channel noise-reduction algorithm to know how and to what extent to modify a specific parameter to improve speech intelligibility. Nevertheless, there are certain cases where it is needed to increase intelligibility based on only one channel. A first scenario is due to the limitation of the hardware, for example when a

microphone has to be as small as possible (forensic applications) or to reduce the cost and the recorded data size.

In order to increase speech intelligibility, some speech modification algorithms have recently been proposed by a numbers of researchers. Some of the algorithms are noise independent. One of these methods is to boost the consonant-vowel power [2][3][4]. Another approach is to deal with tilt flattening and formant enhancement [5]. Manipulating the duration and prosody is also a way to increase speech intelligibility [6]. There are also other strategies like voice conversion [7], and so on.

Some work has been designed on the prior knowledge or estimation of noise, such as the a priori SNR algorithm, the MMSE, the logMMSE, the Wiener filter, and so on. They have all been shown to be ineffective for intelligibility enhancement [8]. The local SNR modification method[9] and the glimpse proportion method [10] were proposed by Tang and Cooke. Another method is cepstral extraction based on the glimpse proportion measure [11]. An optimisation algorithm based on a spectra-temporal perceptual distortion measure [12] has also been proposed. A recently presented method is modifying the differences of subsequent short-term spectrum of speech [13].

There are two main training-based methods which have been studied for intelligibility improvement in different research labs worldwide: the Gaussian mixture model (GMM) [14] based approach and the Deep Neural Network (DNN) [15] based approach.

A GMM is a parametric probability density function represented as a weighted sum of Gaussian components. These components are combined to provide a multimodal density. GMMs are among the most statistically mature methods for clustering. The Expectation Maximization method (EM)[16] is an optimization approach for GMM learning. In statistics, an EM algorithm is an iterative method for finding the maximum likelihood or the maximum *a*

posteriori estimate of parameters in statistical models. GMMs are commonly used for density estimations.

Under some conditions it has been reported that the DNN approach, which better represents the state of the art in machine learning, generalizes better and better processes previously unseen data patterns compared to the GMM method. The main goal of this thesis is to evaluate the performance of the DNN method under different conditions (different types and levels of noise, training mismatch, reverberation).

Since 2006, when Geoffrey E. Hinton proposed an efficient way to train multilayer neural networks [17], a new area of machine learning has emerged, which is called deep learning or deep hierarchical learning [18][19]. The key aspects of machine learning and artificial intelligence have been widened by the techniques developed from deep learning [20][21][22]. There are several active researchers in this area [23]. For speech intelligibility processing using DNN, in 2013, Deliang Wang's group proposed an 85-D feature DNN for speech recognition for hearing-impaired listeners [24]. In 2014, they reported that the Multi-Resolution CochleaGram (MRCG) feature set produces a better result in a multilayer Perceptron, which is a simpler type of neural networks[15].

The main contributions of this thesis are:

- the software implementation of a DNN single channel speech intelligibility algorithm using a combination of publicly available DNN libraries and tools;
- the evaluation of the performance of DNN single channel speech intelligibility algorithm under different conditions such as different types and levels of noise, training mismatch, and reverberation;

- the comparison of the performance of the DNN single channel speech intelligibility algorithm with the GMM single channel speech intelligibility algorithm (whose source code was publicly available), as well as with the DNN approach using other features;
- the thesis was produced as part of a service contract for the Correctional Services of Canada, and the main results of the thesis allowed the client to evaluate the applicability of the speech intelligibility enhancement technology for their needs.

The organization of this thesis is as follows. Chapter 1 (this section) provides an introduction and some background on speech intelligibility improvement and DNN. Chapter 2 mainly focuses on speech intelligibility, on the differences between speech intelligibility and speech quality, and explains some of the reasons why current speech enhancement algorithms do not improve speech intelligibility. Chapter 3 briefly introduces the history of Neural Network learning algorithms. Techniques to robustly and efficiently train a NN are presented in Chapter 4. Chapter 5 introduces the pre-training DNN algorithms. Chapter 6 focuses on the MRCG feature set, and explains how to use this MRCG feature set as a DNN input. Chapter 7 presents the test results evaluating the performance of the DNN speech intelligibility enhancement approach in different situations. Chapter 8 provides a conclusion.

Chapter 2. Speech Intelligibility

The improvement of speech intelligibility in different conditions is the topic of this thesis. This chapter will explain the differences and relationships between speech intelligibility and quality. Some known reasons of why traditional quality enhancement algorithms cannot improve speech intelligibility will be presented. Subjective and objective speech intelligibility assessments will be introduced.

2.1. Speech intelligibility and speech quality

Speech intelligibility and speech quality are different aspects of speech. They are related to each other, but yet they are not equivalent. Speech intelligibility assessment has been widely used to improve the acoustical design of buildings (e.g. airports, railway stations, or shopping malls). Speech quality assessment, on the other hand, has been widely used in traditional and modern telephone networks.

Different assessment methods are used to evaluate these two attributes. In nature, speech quality is quite subjective, because different listeners may have different standards for ‘good’ or ‘poor’ quality. Quality is an assessment of ‘how well’ a speaker produces an utterance. Intelligibility assesses ‘what’ the speaker said, i.e., what words the speaker said and the meaning of the sentence. Intelligibility is not as subjective as quality. It can be measured by a group of listeners identifying the words that they listen to. Intelligibility can be quantified by counting the correct proportion of words that the listeners can identify.

The relationship between speech quality and intelligibility is not fully exposed. Speech with poor quality can be highly intelligible [25], while on the other hand speech of high quality may be totally unintelligible [26].

2.2. Reasons why traditional speech enhancement algorithms do not improve speech intelligibility

Traditional speech enhancement algorithms can improve speech quality but cannot improve speech intelligibility. Until recently, the reasons for this remained unknown (and some of the reasons remain unknown today). There are several reasons that may explain this. The first reason for this is that estimating accurately the background noise spectrum may not be possible. A second factor is that in some cases the distortion introduced by speech enhancement algorithms may be more damaging than the original noise. A third factor is that the non-relevant stochastic modulations arising from the nonlinear noise-speech interactions introduced by the processing can lower the speech intelligibility.

Noise spectrum estimation is needed and is important for most speech enhancement algorithms. It is obvious that the estimation of the background noise spectrum is normally not accurate enough to produce speech intelligibility improvement. Hu and Loizou[27] were able to estimate the noise in car environments more accurately, and have shown that the accurate estimation lead to a small improvement (<10%) in speech intelligibility. This suggests that accurate noise spectrum estimation can improve speech intelligibility. In practice the noise spectrum can never be known very accurately, especially for nonstationary noise. But this is not the only reason why current speech enhancement algorithms cannot improve speech intelligibility. None of the existing algorithms are designed for improving speech intelligibility, as all these algorithms were designed to improve speech quality. They use a cost function to produce the best output, but the cost function does not necessarily correlate well with speech intelligibility. For example, statistical-model based algorithms (MMSE, Wiener filter) utilize the mean-squared error (MSE) to minimize the magnitude spectrum between the clean and estimated spectrum. But MSE does not consider the asymmetric nature of the positive or

negative difference between the clean and estimated spectrum. The positive and negative differences may have the same effect on speech quality, but they have different effects for speech intelligibility. The positive differences would signify attenuation distortion, whereas the negative differences would mean amplification distortion. In speech intelligibility, these two distortions cannot be assumed to be equivalent.

A frequency-domain version of the well-known segmental SNR measure has been introduced to gain a better understanding of the two kinds of distortions mentioned above[27]. This measure can also be called the signal-to-residual spectrum measure SNR_{ESI} . The correlation of SNR_{ESI} with speech intelligibility was found to be 0.81 and the correlation with speech quality was found to be 0.85. The SNR_{ESI} measure can be used in signal-dependent weighting functions computation, which is necessary for predicting intelligibility of the speech with nonstationary noise. The SNR_{ESI} measure is defined below:

$$SNR_{ESI}(k) = \frac{X^2(k)}{\left(X(k) - \hat{X}(k)\right)^2} \tag{2-1}$$

where $X(k)$ is the clean speech magnitude spectrum at bin $\boldsymbol{k}$ and $\hat{X}(k)$ is the magnitude spectrum estimated by a speech enhancement algorithm.

Assume that $D(k)$ is the noise magnitude spectrum. Dividing the numerator and denominator of equation (2) by $D^2(k)$, then we get:

$$SNR_{ESI}(k) = \frac{SNR(k)}{\left(\sqrt{SNR(k)} - \sqrt{SNR_{ENH}(k)}\right)^2} \tag{2-2}$$

where $SNR(k) = X^2(k) / D^2(k)$ is the definition of SNR at bin *k*, and $SNR_{ENH} = \hat{X}^2(k) / D^2(k)$ is the enhanced SNR. Figure 1 shows the relationship between SNR_{ESI} and SNR_{ENH} for SNR = -10dB, 0 dB, and 10dB. When $SNR(k) = SNR_{ENH}(k)$, SNR_{ESI}

becomes infinity. We can see from Figure 1 that the two distortions mentioned above contribute differently to SNR_{ESI}, this means that the same magnitude of these two different distortions doesn't have the same effect on intelligibility. To further understand the contribution of these two distortions, we can divide Figure 1 into 3 regions (relative to the green curve).

- Region I (for green curve, $SNR_{ENH}(k) \le 0$). In this region, we have $\hat{X}(k) \le X(k)$ (or $SNR_{ENH}(k) \le SNR(k)$), with only attenuation distortion.
- Region II (for green curve, $0 < SNR_{ENH}(k) \le 6.02$). In this region, we have $X(k) < \hat{X}(k) \le 2X(k)$ (or $SNR(k) < SNR_{ENH}(k) \le 2SNR(k)$), with amplification distortion up to 6.02dB.
- Region III (for green curve, $SNR_{ENH}(k) > 6.02$). In this region, we have $\hat{X}(k) > 2X(k)$ (or $SNR_{ENH}(k) > 2SNR(k)$), with amplification distortion of 6.02 dB or greater.

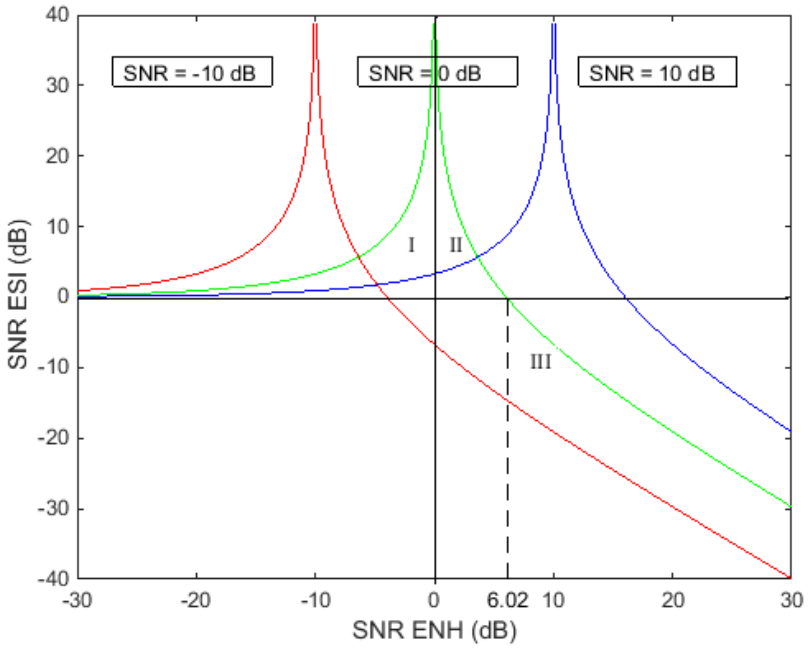


Figure 1 Relationship between SNR_{ESI} and SNR_{ENH} for fixed values of SNR.

From Figure 1 it is clear that in order to get high values of SNR_{ESI}, SNR_{ENH} should fall into Region I or Region II. Region I can be characterized as an attenuation distortion, Region

II can be characterized by low amplification distortion, and Region III can be characterized by large amplification distortion. Loizou and Kim did some tests on 3 different traditional noise enhancement methods to find out how often these three distortions occur [27]. Table 1 shows the results for the Wiener filter and for two spectral-subtractive based speech enhanced algorithms.

SNR Level	Algorithm	Region I	Region II	Region III
0 dB	Wiener	45.33%	14.64%	40.04%
-5 dB		37.73%	12.71%	49.58%
0 dB	RDC_mag	46.86%	15.45%	37.69%
-5 dB		35.88%	14.49%	49.63%
0 dB	RDC_power	36.81%	18.14%	45.05%
-5 dB		28.08%	14.88%	57.05%

Table 1 Percentage of frequency bins falling in the three regions, for three different enhancement algorithms.

Table 1 shows the average percentage of frequency bins falling in each of the three regions. Nearly half of the bins fall into Region I, and nearly half of the bins fall into Region III. The latter can be associated with low SNR_{ESI} values and low intelligibility, thus explaining the weak performance of classic speech enhancement algorithms for speech intelligibility improvement.

To conclude, the entire actual reasons of why traditional speech enhancement algorithms cannot improve speech intelligibility may still be unknown, but there are a few factors that can explain this situation in some way, such as: the actual background noise spectrum is hard to estimate, current speech enhancement algorithms are designed to improve speech quality and this does not necessarily correlate with speech intelligibility, and current speech enhancement algorithms consider positive and negative distortions between clean speech spectrum and estimated speech spectrum to have the same effect, which does not apply to speech intelligibility. To provide speech intelligibility improvement, alternative enhancement methods

have been proposed in the last 3-4 years, which are based on machine learning and data-dependent processing.

2.3. Means to evaluate speech intelligibility

Principally, there are two different methods that can be used to assess speech intelligibility. One is a subjective measure, i.e., an assessment procedure involving speakers and listeners. The other one is an objective measure based on certain parameters that come from the transmission channel.

2.3.1. Subjective intelligibility assessment

The subjective intelligibility measure might be based on phonemes, words (meaningful or nonsense), and sentences. There are several reasons why designing a reliable speech intelligibility test is not easy. First of all, it is not easy to build up a series of tests that represent all major speech phonemes, including relative frequencies. Secondly, to prevent listeners from memorizing all the test material, a list of tests is needed. The difficulties between these different materials should be similar in order to get a fair test, which is not simple. Additionally, due to the fact that words in a sentence can be understood better than just isolated words alone [28], the contextual information should be controlled to be equally intelligible in each test.

During subjective speech intelligibility tests, different techniques are used for the presentation of the test sentence to the subjects and for the type of subject responses. For example, the response method can be either open or closed. An open response requires the listener to provide the words that they heard. A closed response is to provide several options and let the listener choose between them. There are different variations in the scoring method as well.

2.3.2. Objective intelligibility assessment

Subjective tests can provide the most reliable way to assess speech intelligibility, however this kind of method requires a large amount of material, a large number of listeners to cooperate, and are really costly and time-consuming. To overcome these problems, objective measures have been proposed by researchers. The ideal objective assessment should be able to provide with high accuracy the same results as subjective measures. In this thesis, three traditional objective intelligibility assessment measures are discussed: the Speech Intelligibility Index (SII), Speech Transmission Index (STI) and Coherence SII (CSII).

2.3.2.1. Speech intelligibility index

SII is a speech intelligibility measure standard defined by the American National Standards Institute (ANSI), first created in 1969 and later revised in 1997 [29]. SII can be computed by four different methods. They are conceptually the same, and have some difference in the details. The four methods are:

1. Critical frequency band (21 bands)
2. 1/3 octave frequency band (18 bands)
3. Equally-contributing critical band (17 bands)
4. Octave frequency band (6 bands).

The basic concept and steps between these four methods are the same. We will explain how to compute the SII with the second method. The flowchart of the steps involved to calculate the SII is shown in Figure 2. Table 2 shows the list of the frequency bands, the reference internal noise spectrum, the spectrum levels of standard speech for different vocal efforts, and the free-field to eardrum transfer functions for 1/3 octave frequency band.

The SII is calculated with the following steps:

Step 1: In each frequency band i of the 18 bands, measure the equivalent speech spectrum level E_i, the noise spectrum level N_i, and the equivalent hearing threshold level T_i.

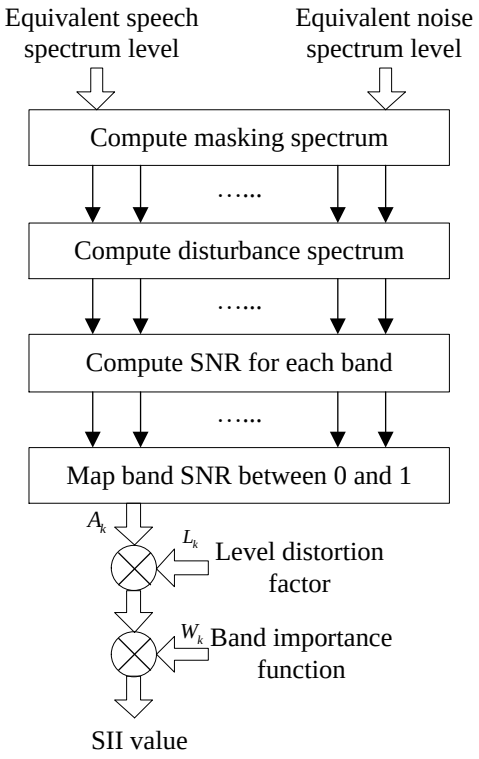


Figure 2 Steps for the computation of the STI measure

In general, E_i can be directly measured at the center of the listener's head, the mid-point between the ears. It requires a large number of talkers to make the measurement stable and accurate. If this situation cannot be satisfied, then the use of free-field to eardrum transfer functions in Table 2 is recommended. For instance, if the sound pressure level at the eardrum of the listener is 30dB at 500Hz, the corresponding free field to eardrum transfer function at 500Hz in Table 2 is 1.80, and the equivalent speech spectrum level E_i is the

difference of the two, i.e. 30-1.80 = 28.2. The noise spectrum level N_i can also be measured at the center of the listener's head.

Frequency band				Standard speech spectrum level for stated vocal effort, dB				Reference internal noise spectrum level dB	Free-field to eardrum transfer function dB
Band no.	Nominal midband freq Hz	Band width	Band importance	Normal	Raised	Loud	Shout		
1	160	15.65	0.0083	32.41	33.81	35.29	30.77	0.60	0.00
2	200	16.65	0.0095	34.48	33.92	37.76	36.65	-1.70	0.50
3	250	17.65	0.0150	34.75	38.98	41.55	42.50	-3.90	1.00
4	315	18.65	0.0289	33.98	38.57	43.78	46.51	-6.10	1.40
5	400	19.65	0.0440	34.59	39.1 1	43.30	47.40	-8.20	1.50
6	500	20.65	0.0578	34.27	40.15	44.85	49.24	-9.70	1.80
7	630	21.65	0.0653	32.06	38.78	45.55	51.21	-10.80	2.40
8	800	22.65	0.0711	28.30	36.37	44.05	51.44	-11.90	3.10
9	1000	23.65	0.0818	25.01	33.86	42.16	51.31	-12.50	2.60
10	1250	24.65	0.0844	23.00	31.89	40.53	49.63	-13.50	3.00
11	1600	25.65	0.0882	20.15	28.58	37.70	47.65	-15.40	6.10
12	2000	26.65	0.0898	17.32	25.32	34.39	44.32	-17.70	12.00
13	2500	27.65	0.0868	13.18	22.35	30.98	40.80	-21.20	16.80
14	3150	28.65	0.0844	11.55	20.15	28.21	38.13	-24.20	15.00
15	4000	29.65	0.0771	9.33	16.78	25.41	34.41	-25.90	14.30
16	5000	30.65	0.0527	5.31	11.47	18.35	28.24	-23.60	10.70
17	6300	31.65	0.0364	2.59	7.67	13.87	23.45	-15.80	6.40
18	8000	32.65	0.0185	1.13	5.07	11.39	20.72	-7.10	1.80
Overall SPL, dB				62.35	68.34	74.85	82.30		

Table 2 1/3-Octave Center Frequencies, Internal Noise Spectrum Levels, Speech Spectrum Levels for Different Vocal Efforts and Free-Field to Eardrum Transfer Function

Step 2: Compute the equivalent masking spectrum

For each band i of the 18 bands, a self-masking spectrum V_i can be computed as

$$V_i = E_i - 24 \tag{2-3}$$

and

$$B_i = \max(V_i, N_i) \tag{2-4}$$

The slope per band for the spread of masking is computed as follows:

$$C_i = -80 + 0.6[B_i + 10\log_{10} F_i - 6.353] \tag{2-5}$$

where F_i is the center-frequency of the 18 bands as shown in Table 2. The equivalent masking spectrum for all bands but the first one is computed as follows:

$$Z_i = 10\log_{10}\left\{10^{0.1N_i} + \sum_{m=1}^{i-1} 10^{0.1[B_m + 3.32C_m 10\log(0.89F_i / F_m)]}\right\} \tag{2-6}$$

Step 3: Compute the disturbance spectrum levels

The equivalent internal noise spectrum is computed as

$$X_i = R_i + T_i \tag{2-7}$$

where R_i is the reference internal noise spectrum as in Table 2. T_i is the hearing threshold level.

$$D_i = \max(Z_i, X_i) \tag{2-8}$$

where D_i is the disturbance spectrum level.

Step 4: Compute the level distortion factors

L_i is the level distortion factor, it can be calculated as follows

$$L_i = 1 - \frac{S_i - U_i - 10}{160} \tag{2-9}$$

where U_i represents the speech spectrum level with normal vocal effort (including raised, loud and shout). L_i is limited to 1 (if L_i is larger than 1, its value is set to 1).

Step 5: Compute the effective band SNR

$$SNR_i = \frac{S_i - D_i + 15}{30} \tag{2-10}$$

where SNR_i is limited with the range of 0-1, the mapping function is shown in Figure 3, and adjusted to account for L_i.

$$SNR_i' = L_i * SNR_i \tag{2-11}$$

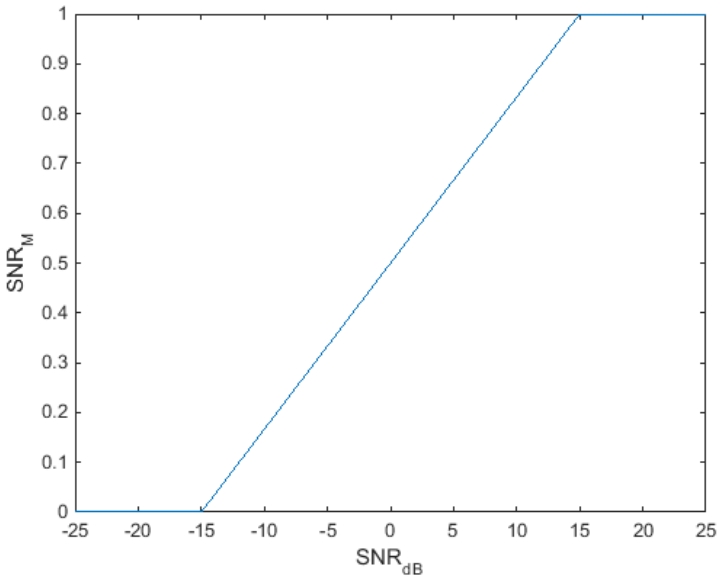


Figure 3 Function used for mapping the SNR to the range of 0–1.

Step 6: Compute the speech intelligibility index

$$SII = \sum_{i=1}^{18} W_i * SNR_i' \tag{2-12}$$

where W_i is the band-importance function shown in Table 2.

SII is good at giving the intelligibility of speech in additive noise or filtered speech. But there are still some limitations and drawbacks. A first one is that SII is not validated for nonstationary noise due to the fact that the algorithm is based on the long-term average spectrum of the noisy speech. A second reason is as issued in the ANSI (1997) standard: the SII measure is not suitable in conditions that contain sharply filtered bands of speech or noise. A third one is that non-linear operations cannot be considered by the SII measure. Methods that try to overcome these issues fall outside the scope of this thesis.

2.3.2.2. Speech Transmission Index

The speech-transmission index (STI) was first presented in the early 1970s by Houtgast and Steeneken [30]. The STI is developed from the Modulation Transfer Function (MTF), which has been used for measuring room effects for auditorium acoustics on speech intelligibility. The steps to compute the STI are similar to the steps to compute the SII. There are some differences on the derivation of the SNR in each band. In the STI computation, instead of 1/3-octave wide bands, octave-wide bands are used. The flowchart to calculate the STI is shown in Figure 4

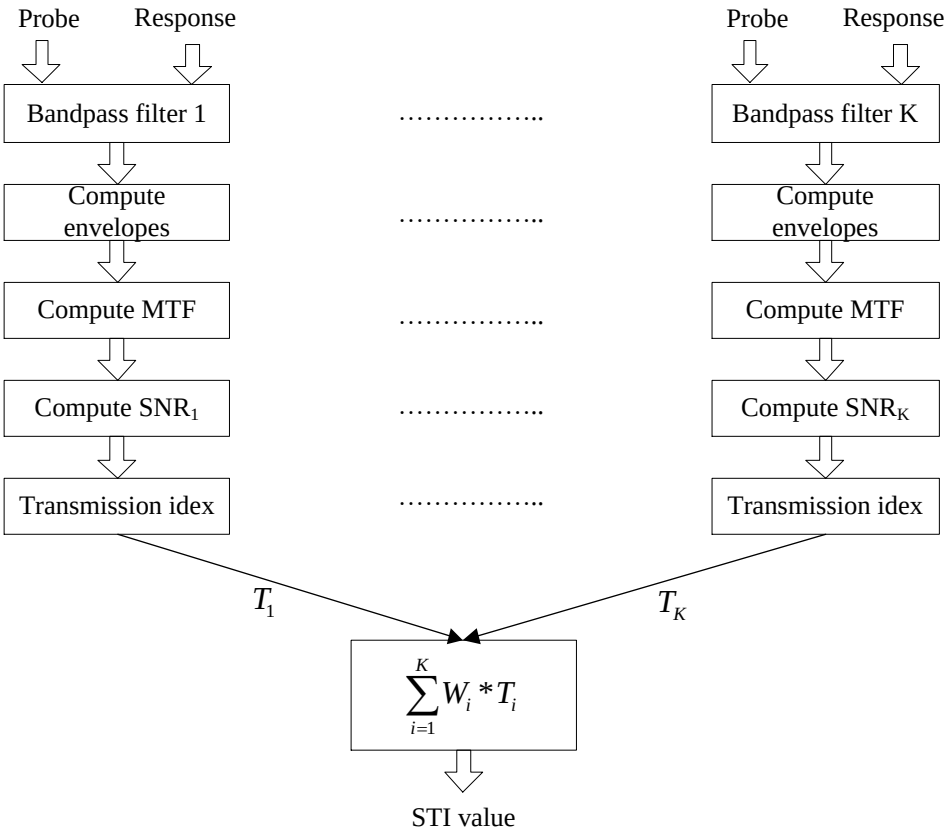


Figure 4 Steps for the computation of the STI measure.

Step 1: Get the probe and response signals

The probe and response signals are divided into K bands by bandpass filtering. The center frequencies of the octave bands are within 125 and 8000Hz.

Step 2: Computation of band envelopes

Apply a Hilbert transform or full-wave rectification and a low pass filter on the input data. The envelopes are down-sampled, in order to reduce the bandwidth and make the design of the modulation filters easier.

Step 3: Computation of MTF

The MTF is the ratio of input and output modulation spectrum, hence the MTF can be defined as:

$$MTF(f) = \alpha \sqrt{\frac{P_{yy}(f)}{P_{xx}(f)}} \tag{2-13}$$

where α is the ratio of the mean of the input envelope μ_x and the mean of the output envelope μ_y, which means $\alpha = \frac{\mu_x}{\mu_y}$. P_{xx} is the modulation spectrum of the input envelope. P_{yy} is the modulation spectrum of the output envelope. Typically, the MTF is computed for modulation frequencies f between the range of $f = 0.63Hz$ to $f = 12.7Hz$ under the rule of 1/3-octave filter spacing.

Step 4: Computation of SNR

In each band i, the SNR is calculated for modulation frequency f as follows:

$$SNR_i(f) = 10\log_{10}(\frac{MTF(f)}{1 - MTF(f)}) \tag{2-14}$$

Similar to the SII, the SNR here is also limited within the range -15 to 15dB, and then averaged over all modulation frequencies f to get SNR_i' in the i th band.

Step 5: Computation of transmission index (TI)

The TI_i transmission index is then:

$$TI_i = \frac{SNR' + 15_i}{30} \tag{2-15}$$

Step 6: Computation of STI

The STI is the weighted sum of TI_i:

$$STI = \sum_{i=1}^{K} W_i * TI_i \tag{2-16}$$

where, W_i is the band-importance weights ($\sum W_i = 1$)

In [31], seven octave-wide bands were used, spanning from 125 Hz to 8000Hz, $W_i = [0.13,\ 0.14,\ 0.11,\ 0.12,\ 0.19,\ 0.17,\ 0.14]$.

Like the SII, the STI measure has some limitations too. The STI is not good to deal with non-linear processes (e.g. dynamic envelope compression in hearing aids). STI processing may introduce additional modulation, which the STI measure itself can treat as increased SNR, leading to a false prediction of speech intelligibility.

2.3.2.3. Coherence SII (CSII)

The STI measure makes use of the modulation envelope. Instead, the CSII utilizes the coherence between the clean and enhanced (or target) signals to compute the SNR in each band. The CSII method uses the magnitude-squared coherence (MSC) to calculate the intelligibility index.

Let the target speech spectrum be denoted as $Y(\omega)$, then

$$Y(\omega) = X(\omega) + N(\omega) \tag{2-17}$$

where $X(\omega)$ is the clean speech spectrum, and $N(\omega)$ is the noise signal spectrum.

The MSC between $Y(\omega)$ and $X(\omega)$ can be calculated as

$$\left|\gamma_{XY}(\omega)\right| = \frac{\left|P_{XY}(\omega)\right|}{P_{XX}(\omega)P_{YY}(\omega)} \tag{2-18}$$

where $P_{XX}(\omega)$, $P_{YY}(\omega)$ are power spectrum densities of the signals $x(n)$ and $y(n)$. $P_{XY}(\omega)$ is the cross power spectrum density, with $P_{XY}(\omega) = E[X(\omega)Y^H(\omega)]$ $P_{XX}(\omega) = E[X(\omega)X^H(\omega)]$. The superscript ^{H} means conjugate.

In real life, the MSC can be calculated by dividing the signals into M windowed frames, sometimes overlapping between neighbor frames, and averaging over all M frames. Then the MSC is given as :

$$\left|\gamma_{XY}(\omega)\right|^2 = MSC'(\omega) = \frac{\left|\sum_{i=1}^{M} X_i(\omega)Y_i^H(\omega)\right|}{\sum_{i=1}^{M}\left|X_i(\omega)\right|^2 \sum_{i=1}^{M}\left|Y_i(\omega)\right|^2} \tag{2-19}$$

where $MSC'(\omega)$ is the estimated MSC, M is the total number of frames, and the subscript i is for the number of each frame.

The CSII is a SII based intelligibility index, which means that the SNR in equation (2-10) is computed as follows:

$$SNR_{ESI}(j,i) = \frac{\sum_{k=1}^{N} G_j(\omega_k)MSC'(\omega_k)\left|Y_i(\omega_k)\right|^2}{\sum_{k=1}^{N} G_j(\omega_k)[1-MSC'(\omega_k)]\left|Y_i(\omega_k)\right|^2} \tag{2-20}$$

in which N is the number of FFTs and $Y_i(\omega_k)$ is the Fourier spectrum of the target signal in the i th frame. $\sum_k G_j(\omega_k)\left|Y_i(\omega_k)\right|^2$ integrates the power within the j th band.

The function $G_j(f)$ can be calculated as follows:

$$G_f(f) = (1+p_j g)e^{-p_j g} \tag{2-21}$$

where $p_j = 4000 * q_j / b_j$, $g = \left|1 - f / q_j\right|$, b_j is the bandwidth, q_j is the center frequency of the j th band.

A mel-frequency division, 16 band $G_f(f)$ is shown in Figure 5. As in the SII, SNR_{ESI} is also limited to the -15 to 15 dB range, and mapped to [0, 1] by the function shown in Figure 3. The output after these functions is denoted as $T_{CSII}(j,i)$. Then the CSII is:

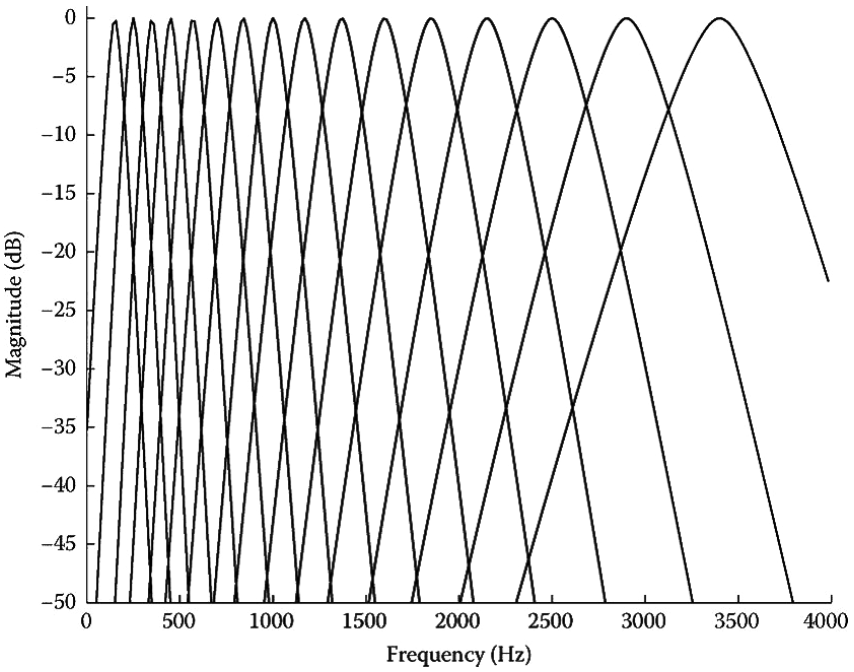


Figure 5 Frequency response of $G_f(f)$

$$CSII = \frac{1}{M}\sum_{i=0}^{M-1}\frac{\sum_{j=1}^{K} W(j) T_{CSII}(j,i)}{\sum_{j=1}^{K} W(j)} \tag{2-22}$$

where K is the total number of bands and $W(j)$ is a band-importance function.

Three CSII-based measures can be derived by separating the M speech frames into 3 level regions [32]. Denote the overall root-mean-square level of the whole utterance as RMS, the region at and above RMS consists mostly of sonorant segments (vowels, semivowels, and nasals), and this region is called a high-level region. The mid-level region ranges from RMS to RMS-10dB, and the low-level range is from RMS-10dB to RMS-30dB.

A linear combination of the three CSII values followed by a logistic function transformation was subsequently used to model the intelligibility scores, and the resulting intelligibility measure was called I_3.

$$c = -3.47 + 1.84 * CSII_{Low} + 9.99 * CSII_{Mid} + 0.0 * CSII_{High}$$
$$I_3 = \frac{1}{1+e^c} \tag{2-23}$$

Of all the measures described above, the CSII performs the best [8]. The correlation coefficients between sentence intelligibility scores and objective intelligibility measures are [8]: the high-level CSII is at 0.85, the mid-level CSII is at 0.91, and the low-level CSII is at 0.86, which are relatively higher than the other measures studied in that article. The I_3 measure produces a 0.92 correlation, which is the highest and is maintained for all noise types. Objective intelligibility assessments as shown here may be suitable to estimate how intelligible a speech file is, but unfortunately they can't give the information about how close the processing is to the ideal IDBM mask processing, and to what extend the processing can be improved. The HIT-FA method will be used to evaluate the estimated mask compared to the IDBM, and it will be further discussed in Chapter 7.

2.4. Ways to improve speech intelligibility

There are several ways to improve speech intelligibility. SNR-based channel-selection algorithms require the estimation of the SNR for making a decision in each Time-Frquency (T-F) bin: retain or discard. There are classic algorithms to estimate the SNR of the signal instantaneously such as the so-called decision-directed approach. In order to improve speech intelligibility, firstly we need to control the distortions introduced by noise-reduction algorithms; secondly, we need to increase the overall SNR so as to improve the intelligibility. Channel-selection based algorithms have been shown to work well in improving intelligibility

under favorable conditions. Channel-selection based algorithms work as follows: decompose the signal into short-time frames and perform frequency analysis on each frame, leading to time-frequency (T-F) bins. A channel-selection based algorithm will retain the T-F bins where speech is dominant, and discard the T-F bins where noise is dominant. The algorithm makes the retain/discard decision based on the SNR of each T-F bin, compared to a threshold called the local SNR criterion (LC). In an ideal case, where access to clean speech and noise-only signals is possible and the exact SNR can be computed, using a channel-selection algorithm we can get the ideal binary mask (IDBM):

$$B(k,t)=\begin{cases}1 & SNR>LC\\ 0 & otherwise\end{cases} \tag{2-24}$$

where $B(k,t)$ is the IDBM mask at channel (frequency) k and time t.

The IDBM has been shown to improve the speech intelligibility at any input SNR level (even as low as -40 dB). The IDBM is an unrealistic condition that can never occur in real life, but for several reasons this result is extremely important. In particular, the outcome of the IDBM can provide an upper bound, so that the IDBM can be a criterion to estimate the performance of a practical algorithm.

As previously mentioned, data-dependent machine learning based methods such as DNN and GMM have recently been reported to be effective in improving speech intelligibility. Such training-based methods use a relatively large corpus of speech and noise sources, together with the calculated IDBM as the input of a system to be trained.

A typical learning based algorithm is shown in Figure 6 below.

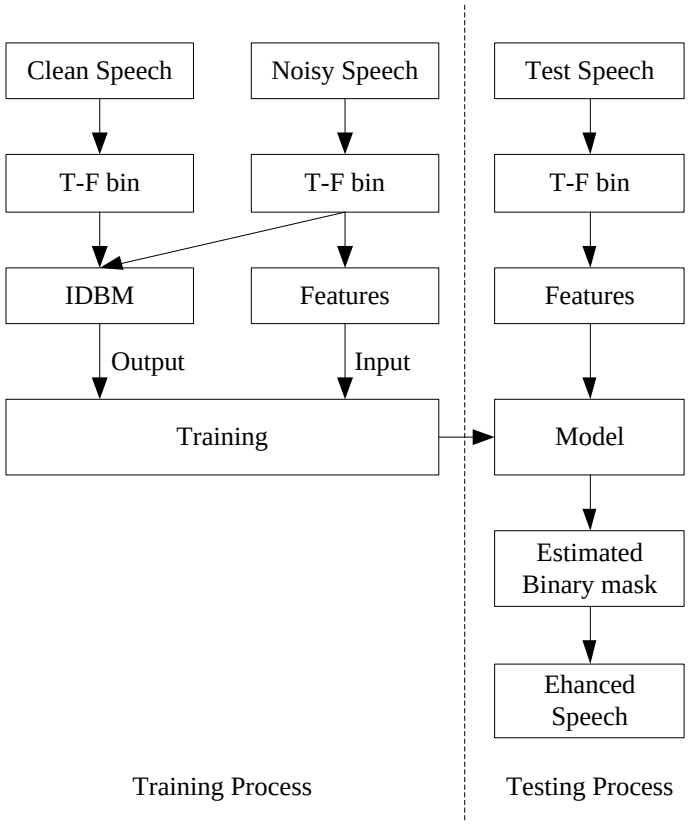


Figure 6. Overview of typical learning based algorithm

Once the system has been trained and a model is obtained, the model is then used to estimate the binary mask when processing new data (noisy files), without knowledge of the speech or noise-only signals. As will be explained later in more details, this is the approach of the DNN method to be implemented in this thesis.

Chapter 3. Neural Network

There are problems that cannot easily be formulated analytically or calculated exactly, such as to recognize a three dimensional object from a new angle, with a new lighting shade, or with different surface colors. These tasks are very hard for computers to accomplish, but fairly easy for the human brain to achieve. This is why artificial Neural Networks (NN) have been developed and used. NN are inspired by biological neural networks and are used to estimate or approximate functions that are unknown and can depend on a large number of inputs [33]. A Deep Neural Networks (DNN) is a NN with multiple hidden layers (usually at least two hidden layers) of units between the input and output layers[34]. NN are discussed in this chapter. The learning algorithm for NN is also introduced, which is the key process for the use of NN.

3.1. Some neural network history

3.1.1. Linear neuron and Perceptron

A linear neuron (which can also be called a linear filter) is a system that with an input-output behavior that can be described in terms of a linear function. The linear neuron is one of the simplest type of neurons which removes all complicated details, but still allows us to apply mathematics to make analogies with familiar systems. This can help us understand how neural networks learn. A linear neuron can be described as follows:

$$y = b + \sum_i x_i w_i \tag{3-1}$$

where x_i is the i th input. w_i is the i th weight on the connection between output y and the i th input. b is the bias. $\sum_i$ means sum over all x_i value. Each input symbolizes a neural node, and the weighted sum plus the bias represent the neuron output.

The Perceptron was first described by Rosenblatt [35] . It was the first generation of neural networks. A Perceptron is similar to linear neurons. The difference between a Perceptron and linear neurons is that the output unit of a Perceptron adds a decision unit at the output of the linear neuron. A Perceptron can be defined as follows:

$$z = b + \sum_i x_i w_i$$
$$y = \begin{cases} 1 & z \geq 0 \\ 0 & otherwise \end{cases} \tag{3-2}$$

The bias b here is equivalent to set the output to y by a threshold decision function. The bias b can also be treated as a weight. Figure 7 shows a typical layout of Perceptron.

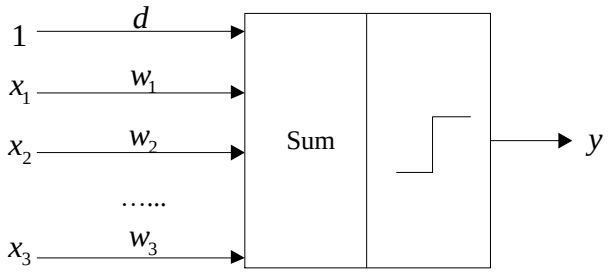


Figure 7. Perceptron layout

Assume the number of output nodes is N_{in} . Then the Perceptron can be defined as:

$$z = b + \sum_i^{N_{in}} x_i w_i = Wx$$
$$y = \begin{cases} 1 & z \geq 0 \\ 0 & otherwise \end{cases} \tag{3-3}$$

In which, $W = [b, w_1, w_2, ..., w_{N_{in}}]$, $x = [1, x_1, x_2, ..., x_{N_{in}}]^T$, where $[\ ]^T$ means transpose.

3.1.2. Limitation of the Perceptron

The Perceptron is good at solving classification problems that are linearly separable. Figure 8 shows an example of linearly separable data set in a two dimensional plane. To better understand the meaning of it, nonlinear separable and inseparable examples are also provided.

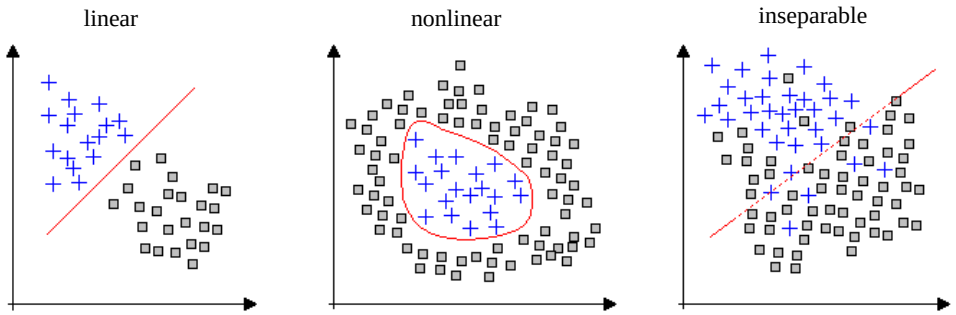


Figure 8. 2 dimension linearly separable, nonlinear separable, and inseparable

The Perceptron has some limitations when it comes to nonlinear separation problem, as it cannot find the proper weights to solve the problem. Take an XOR gate function as an example. The XOR gate is a digital logic gate, if one and only one of the two input is one, then the output is true. If both inputs are 0 or 1, then the output is false. The XOR gate function truth table is shown below:

Input		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table 3 XOR truth table

The Perceptron layout for this function can be shown in Figure 9a:

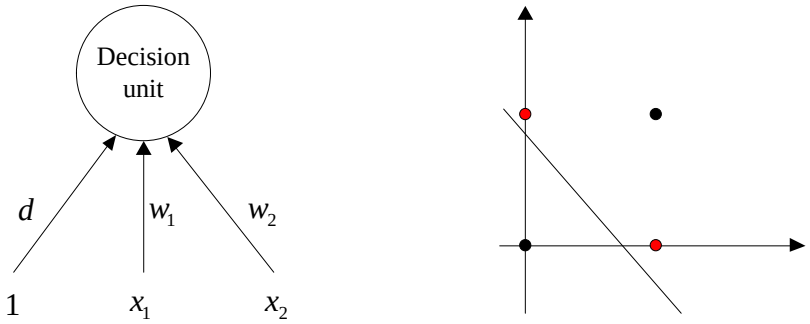


Figure 9. a) Perceptron layout for XOR function b) XOR gate function

Given the 4 input-output pairs in Table 3, we can have 4 inequalities:

$$\begin{aligned} d+0*w_1+0*w_2 \le 0 \quad &\Rightarrow \quad d \le 0 \\ d+0*w_1+1*w_2 \ge 0 \quad &\Rightarrow \quad w_2 \ge -d \\ d+1*w_1+0*w_2 \ge 0 \quad &\Rightarrow \quad w_2 \ge -d \\ d+1*w_1+1*w_2 \le 0 \quad &\Rightarrow \quad w_1+w_2 \le -d \end{aligned} \tag{3-4}$$

These four inequalities are impossible to satisfy, which means that the Perceptron cannot solve this simple problem. This illustrates the defect of the Perceptron. The visible proof is shown in Figure 8b, where we see that the problem is not linearly separable.

3.1.3. Development of the Perceptron

To overcome the defect discussed above, a series of strategies were adopted, making the Perceptron become a true NN. One of the changes was to give the Perceptron an activation function, as shown below:

$$\begin{cases} z = b + \sum_i x_i w_i \\ v = f_{act}(z) \\ y = f_{out}(v) \end{cases} \tag{3-5}$$

where f_{act} is the activation function, f_{out} is the output function, and v is the activation vector (input). f_{act} and f_{out} are usually fused together, to make the learning

algorithm more simple. There are a lot of possible activation and output functions, transforming the Perceptron into different neurons. The most used neurons are the rectified linear neurons [36], sigmoid neurons [37], hyperbolic tangent neurons [38], and softmax neurons [39].

1. Rectified linear neurons

Rectified linear neurons compute a linear weighted sum of inputs, and the output is a non-linear function of the total input. The activation function is as follows:

$$v = \begin{cases} z & z \geq 0 \\ 0 & otherwise \end{cases} \tag{3-6}$$

The output function is simple:

$$y = v \tag{3-7}$$

The graph of a rectified linear function is:

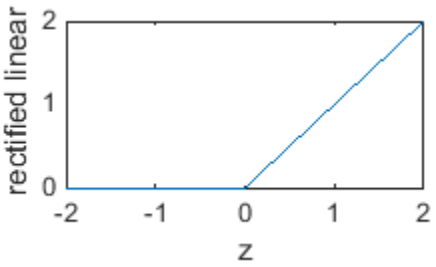


Figure 10. Rectified linear function

2. Sigmoid neurons

These kind of neurons give a real-valued output that is a smooth and bounded function of the inputs. They use logistic function as activation function, which has nice derivatives that makes the learning procedure easier. The activation function is as follows:

$$v = \frac{1}{1+e^{-z}} \tag{3-8}$$

The output function is also simple:

$$y = v \tag{3-9}$$

The graph of a sigmoid function is:

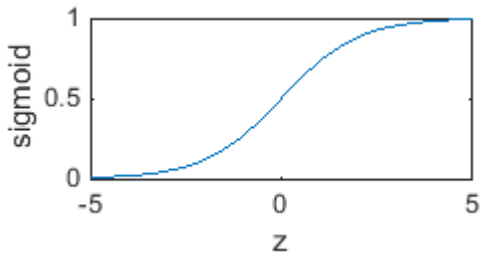


Figure 11. Sigmoid function

3. Hyperbolic tangent neurons

The hyperbolic tangent function is a rescaled version of the sigmoid function. The difference between the hyperbolic tangent neuron and the sigmoid neuron is just the output range. For the hyperbolic tangent neuron it is [-1,1], while for the sigmoid it is [0,1].

The activation function is as follows:

$$v = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3-10}$$

The output function is also simple:

$$y = v \tag{3-11}$$

The graph of a hyperbolic tangent function is:

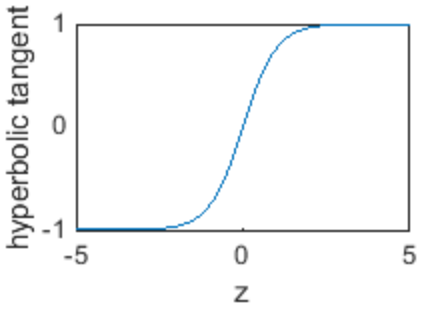


Figure 12 hyperbolic tangent function

4. Softmax neurons

The softmax neuron is typically used in multi-class classification tasks, where each output node represents a class. Assume there are a total of N_{out} classes to separate. The output of the activation function represents the probability of each node. The output function selects the highest probability, and set that node to one, then sets the others to be zero.

The activation function is as follows:

$$v_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3-12}$$

where v_i is the i th vector of v, e^{z_i} is the i th vector of e^z.

The output function is also simple:

$$y_i = \begin{cases} 1 & if \quad v_i = \max\{v_1, v_2, \ldots, v_{N_{out}}\} \\ 0 & otherwise \end{cases} \tag{3-13}$$

where y_i is the i th vector of y.

In addition to have activation and output functions as mentioned above, there are other ways to make NN more nonlinear. One of them is adding hidden layers, which creates a typical feedforward NN [38]. Adding more hidden layers can make the computational complexity of the learning algorithms increase exponentially, and such structures and training have only become possible with the calculation abilities of modern computers.

Denote the input layer as layer 0, and the output layer as layer L. Then in the first L layers:

$$v^l = f_{act}(z^l) = f_{act}\left(W^l v^{l-1}\right) \quad (for\ 0 < l < L) \tag{3-14}$$

in which $v^0 = x$, W^l is the weight of each layer. $z^l = \{z_1^l, \ldots, z_{N_L}^l\}$. Assume that j is a node in layer L, i is a node in layer $L-1$, then $z_j^l = \sum_{i=0}^{N_{L-1}} w_{ij}^l v_i^{l-1}$, N_L is the number of neuron in layer L, and N_{L-1} is the number of neuron in layer $L-1$.

Then the output y is:

$$y = f_{out}(v^L) \tag{3-15}$$

Figure 13 is a graph showing multiple hidden layers in a simple NN.

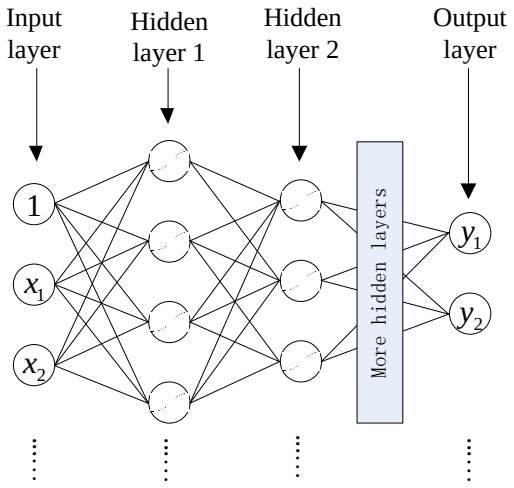


Figure 13 A simple NN with multiple hidden layers

The recurrent neural network is another kind of method to break the linearity problem of the Perceptron. Recurrent neural networks add connection between hidden nodes. This kind of neural network has been widely used, but is out of the scope of this thesis.

Since the backward propagation of error learning rule was developed and published in the 1980s [40][41], the development of the neural network field has almost been explosive.

Numerous type of NNs have been developed, and they have been utilized in hundreds of different research and industry areas.

3.2. Neural network learning algorithm

Learning (or training) here means a method such that the network can adapt its weights in order to achieve the desired behavior. The whole process of learning is to present an amount of desired input-output examples (normally the more the better) to the network, and then the network can make some corrections to its weights according to these examples. Learning algorithms can be divided into three main groups, *supervised*, *unsupervised,* and *hybrid* methods.

Supervised learning is a kind of discriminative learning method where the input and the desired output are all known to the network. The network adjusts its weights by using the deviation of the output from the desired signal. Supervised learning can be also called 'learning with teachers'.

Unsupervised learning is a generative learning method where only the input data is known to the network, and the learning algorithm itself finds the different links between the samples and attempts to classify/separate them. In real life, a person can learn things without a teacher, e.g. when a person has seen a bunch of tea cups, when he/she being presented a new different tea cup, he/she can instantly know that it is a tea cup. Unsupervised learning is trying to simulate this process.

Hybrid learning a third category used in Deep Neural Networks, which makes use of both discriminative and generative learning. This kind of training method usually works as follows: let the network learn in an unsupervised fashion to find good initialization points

for highly nonlinear parameter estimation problems (pre-training process), and then perform supervised learning starting from that initialization point, for fine tuning.

The 'Backward propagation of errors' (Backpropagation) is a common method of supervised learning method. Several methods have been developed to help backpropagation maintain some robustness. But backpropagation alone still has its limitations when it comes to deep architecture networks.

3.2.1. Backpropagation

The typical backpropagation method uses gradient descent as its optimization method, which calculates the gradient of a loss function with respect to all the weights in the network. The gradient is applied first at the last hidden layer until the first layer, which is called backward processing. The loss function requires a known, desired output for each input, so this learning process is supervised. Backpropagation utilize the calculus chain rule to iteratively compute the gradients for each layer and neuron. Thus the activation function of the neurons should be differentiable. In this section, we show how to use backpropagation with the last three types of neurons previously mentioned.

To get a better understanding of backpropagation, the delta rule is briefly derived here. The delta rule uses a linear activation function. The loss function in this case is defined as the mean squared error (MSE) of all training cases.

$$E = \frac{1}{M}\sum_{n=1}^{M}\frac{1}{2}\| t^n - v^n \|^2 \tag{3-16}$$

in which M is the total number of training cases, the superscript ^{n} means the n th training case, t is the desired output, and y is the linear activation function as $v = \sum_i x_i w_i$. If the output is not of one dimension, the Euclidean norm is applied.

Then according to the chain rule, for the i th weight, the error derivative for the weight can be written as:

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_n \frac{\partial v^n}{\partial w_i}\frac{dE}{dv^n} = -\frac{1}{M}\sum_{n=1}^{M} x_i^n (t^n - v^n) \tag{3-17}$$

There are 3 ways to apply the error derivative to the weights, namely batch learning, online learning, and mini-batch learning.

The batch delta rule changes the weights in proportion to their error derivatives summed over all training cases:

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i} = \varepsilon \frac{1}{M}\sum_{n=1}^{M} x_i^n (t^n - v^n) \tag{3-18}$$

where ε here is usually called the learning rate, or the step size.

The online learning rule changes the weight in proportion to their error derivatives based on each single training case:

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i} = \varepsilon x_i (t - v) \tag{3-19}$$

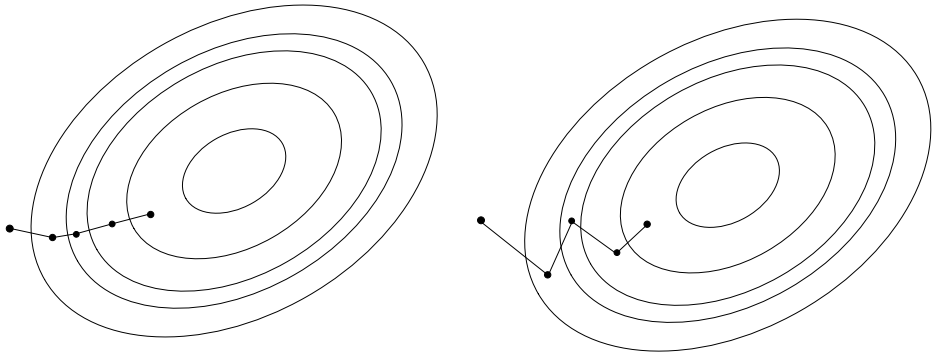
The mini-batch learning rule is the combination of the batch learning and online learning together. All the training cases are separated into several portions (mini-batch), and within each portion a batch learning is performed with the sum of the error derivatives:

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i} = \varepsilon \frac{1}{M'}\sum_{m=1}^{M'} x_i^m (t^m - v^m) \tag{3-20}$$

where m is the index within each mini-batch, and M' is the size of the mini-batch.

Before explaining the difference between these 3 learning ways, the error surface should be mentioned. The error surface of a linear neuron with a squared error loss function is a quadratic bowl. For a 3 dimension error surface, the vertical cross-sections are

parabolas, and the horizontal cross-sections are ellipses. For multi-layer, non-linear neural networks, the error surface is much more complicated. Batch learning performs steepest descent on the error surface. The learning trajectory travels perpendicular to the contour lines, as shown in Figure 14 (a). Online learning zig-zags around the descent direction of the batch learning lines, as shown in Figure 14 (b). The problem with batch learning is that if the training cases are too many, the learning algorithm may have to wait too long before each weight updates. On the other hand, online learning is a noisier learning process compared to batch learning, and it can sometimes show slow convergence properties. Mini-batch learning can provide a good trade-off between these two methods, which is the reason why mini-batch learning is used in this thesis.



(a) (b)

Figure 14 Simplest bacth learning (a) and simplest online learning (b)

Backpropagation is a generalization of the delta rule which is similar to the Perceptron training rule, although the derivation is different. For different activation functions, unlike the linear activation functions used in the delta rule of equation (3-5), the derivative of the output with respect to each weight (equation (3-17)) can be written as (where for a neat expression we omit the average over samples: $\frac{1}{M}\sum_{m=1}^{M}$ or $\frac{1}{M'}\sum_{m=1}^{M'}$):

$$\frac{\partial E}{\partial w_i} = \frac{\partial v}{\partial w_i}\frac{dE}{dv} = \frac{\partial z_i}{\partial w_i}\frac{\partial v}{\partial z_i}\frac{dE}{\partial v} = x_i df_{act}(t-v) \tag{3-21}$$

where df_{act} is the derivative of the activation function, $\frac{\partial z_i}{\partial w_i} = x_i$, $\frac{\partial v}{\partial z_i} = df_{act}$, and $\frac{dE}{\partial v} = (t-v)$.

Depending on the choice of the loss function, different derivatives are used. For the MSE loss function, the sigmoid neuron uses the sigmoid function as its activation function (3-8), and the derivative of the sigmoid function is shown below:

$$df_{act} = \frac{dv}{dz} = \frac{d(1+e^{-z})^{-1}}{dz} = \frac{-1(-e^{-z})}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}} = v(1-v) \tag{3-22}$$

Then equation (3-21) for sigmoid neurons can be written as:

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_n x_i^n [v(1-v)](t^n - y^n) \tag{3-23}$$

The derivative of the hyperbolic tangent neuron is similar to the sigmoid neuron. As for softmax neurons, instead of using the MSE loss function the cross-entropy (CE) is normally used as the loss function:

$$E = \frac{1}{M}\sum_{n=1}^{M}\left(-\sum_{i=1}^{N_L} t_i \log v_i\right) \tag{3-24}$$

Since this is a classification problem, only one component of t is 1, and others are all 0. Then E becomes a simple equation. Again omitting $\frac{1}{M}\sum_{m=1}^{M}$ or $\frac{1}{M'}\sum_{m=1}^{M'}$ we can write:

$$E = -\log v_c \tag{3-25}$$

in which subscript $_c$ is the non-zero output dimension.

If the activation function is the softmax function in (3-12), then $\frac{\partial v_i}{\partial z_i} = v_i(1-v_i)$ (same form as the sigmoid activation function derivatives), and according to the chain rule:

$$\frac{\partial E}{\partial z_i} = \sum_j \frac{\partial E}{\partial v_j}\frac{\partial v_j}{\partial z_i} = y_i - t_i \tag{3-26}$$

in which, $\sum_j$ means adding all the elements in the group.

Then

$$\frac{\partial E}{\partial w_i} = x_i(y_i - t_i) \tag{3-27}$$

So far, the error for the last layer of the neural network has been calculated as $\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i}$. The approach to calculate errors in multilayer networks (as shown in(3-14)) is similar. Next we explain how to calculate the error in layer $L-1$ using the error calculated from layer L.

Suppose the errors in layer L are already calculated as: $\frac{\partial E}{\partial v^l}$, then we have:

$$\begin{gathered}
\frac{\partial E}{\partial z_j^l} = \frac{dv_j^l}{dz_j^l}\frac{\partial E}{\partial v_j^l} = \frac{df_{act}}{dz}\frac{\partial E}{\partial v_j^l} = v_j^l(1-v_j^l)\frac{\partial E}{\partial v_j^l} \\
\frac{\partial E}{\partial v_i^{l-1}} = \sum_{j=1}^{N_L}\frac{dz_j^l}{dv_i^{l-1}}\frac{\partial E}{\partial z_j^l} = \sum_{j=1}^{N_L} w_{ij}^l \frac{\partial E}{\partial z_j^l} \\
\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial z_j^l}{\partial w_{ij}^l}\frac{\partial E}{\partial z_j^l} = v_i^{l-1}\frac{\partial E}{\partial z_j^l}
\end{gathered} \tag{3-28}$$

Then the error in layer $L-1$ can be calculated using $\frac{\partial E}{\partial v^{l-1}}$, continuing on for layer $L-2$ until layer 1.

No matter whether using batch, mini-batch or online training algorithms, all the samples may need to be used multiple times. Each time where the process goes through all the samples is called one iteration. To get a desirable network model, the network may need to be trained for hundreds or even thousands of iterations.

3.2.2. The ups and downs of backpropagation

Backpropagation can do a good job for learning some non-linear features. But by the late 1990's most machine learning researchers had given up on it, because it failed to perform well for highly non-linear multiple hidden layer networks. Some conjectures were proposed as to why it failed, such as it could not make good use of multiple hidden layers, or it did not work well in recurrent network / deep auto-encoders, or Support Vector Machines (SVM) worked better but with much fancier theory.

Compared to today, computers were thousands of times slower back then, labeled datasets were much fewer, and deep networks were not initialized sensibly, which was responsible for the failure of backpropagation in "deep" neural networks (with several layers). In deep networks, the gradients tend to die, so the network training is easily stuck in local minima or plateaus, because the initial weights are typically far away from the global minima. This issue together with poor computer speed and lack of labeled datasets prevented backpropagation from being successful in training deep neural networks. Recent insight pointed out by Hinton *et al.* [18] using a greedy layer-wise unsupervised learning procedure, i.e., initializing the NN with a Restricted Boltzmann Machine, has proved to work well in training DNN. This is further described in later chapters.

Chapter 4. Training Techniques

It is important to prevent problems like overfitting in order to get final reliable DNN models. And it is essential to make the training process converge quickly to save training time. These problems are discussed in this chapter.

4.1. Overfitting and preventing overfitting

One of the problems that neural networks (and other models) suffer from is overfitting. The training data is a set of discontinuous samples, which contains regularities in the mapping from input to output. It also contains other noises, such as unreliable target values, and sampling error. The sampling error would cause accidental regularities because particular training cases have been used. When a learning algorithm fits the model for a NN, it cannot tell the real regularities from the ones that come from sampling error. Training algorithms would try to fit both kinds of regularities. This would cause the NN to produce a wrong model.

Figure 15 shows an example of normal fitting an overfitting. The red line fits all the data better than blue line. But we should give more credits to the blue line, simply because the red line models the accidental regularities from sampling errors.

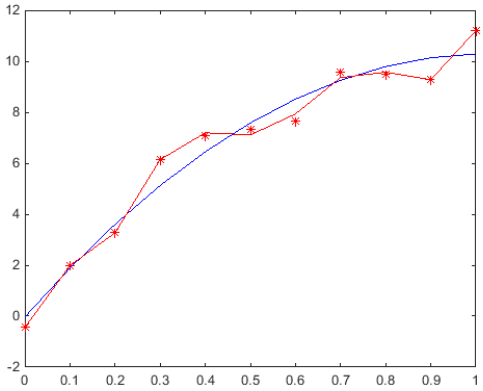


Figure 15 An example of overfitting

There are several ways to reduce overfitting. The first thing we can do to prevent overfitting is get more data. If the computer calculation ability is powerful enough to handle the amount of data, getting more data is always a good way to reduce this effect. A second thing we can do is try to find a model with the just right modeling capacity. Take the overfitting problem shown in Figure 15 as an example. If somehow we know that the desired polynomial order is 2, and then we limit the neural network model to order 2, overfitting could be prevented. There are several ways to limit the modeling capacity of a NN. Controlling the architecture, early stopping, weight-decay, and adding noise are possible ways to do the limitation. A third thing that could be helpful is to get many different models, and average them. Using different forms of models, or training the model on different subset of the data can help to generate different models. A fourth thing to try to get rid of overfitting is a Bayesian approach, which is using a single neural network architecture but averages the predictions made by different weight vectors. A combination of the above methods is often used in NN training.

An efficient way of choosing parameters/weights in NN is cross-validation. Cross-validation divides all the training data into three subsets, namely training data, validation data and test data. Training data is used for learning. Validation data is not used directly for learning, but is used for deciding which parameters work best. Test data is used to get the final estimate of how well the network works.

Weight penalty is a way of applying weight-decay. Standard weight penalty involves adding an extra term to the cost function that penalizes the squared weights. There are two common weight penalties, $L1$ and $L2$:

$$\begin{aligned} \textit{weight decay}\,\mathrm{L1} &= E + \lambda \sum_i \left| w_{ij} \right| \\ \textit{weight decay}\,\mathrm{L2} &= E + \lambda \sum_i w_{ij}^2 \end{aligned} \tag{4-1}$$

in which E is the cost function, w_{ij} is the weight of each layer, and λ is a constant for the penalty. Weight penalty prevents the network from using weights that it does not need. It makes a smoother model in which the output changes more slowly as the input changes.

4.2. Dropout

Dropout is a training technique that averages many large neural nets and at the same time deals with dropout. The key idea is to randomly drop units (along with their connections) from a neural network during training. This prevents the units from co-adapting too much. Dropping units creates thinned networks during training [42]. Averaging several models is a typical way to improve performance of NN. However, training many different NN models is impractical, especially when the NN are large.

Dropout is a way that discards neural nodes which can be in hidden and input layers. The discarding process happens purely randomly. Figure 16 shows a dropout version of a NN (originally shown in Figure 13). As shown in Figure 16, the dropout NN is a pruned network from Figure 13. If the dropout rate is 0.5, it means dropping half of all the units in a NN. Then in a neural net with n nodes, there will be 2^n possible pruned neural networks. All these NNs share weights, so the number of parameters remain still. Training a dropout NN can be seen as training a set of 2^n pruned NNs with extensive weight sharing. Sharing weights means that every model is very strongly regularized. This is a better way to regularize than using the $L1$ and $L2$ penalties that pull the weights towards zero [43]. During the testing stage, a dropout NN computes the geometric mean of the predictions from all 2^n models.

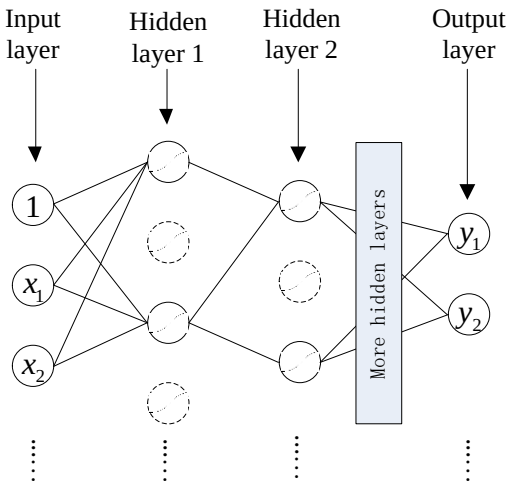


Figure 16 A dropout NN

4.3. Methods to speed up mini-batch learning

To make the training process convergence fast is important in training, sometimes it may even avoid the chance to be stuck in a local plateau in the loss function. There are several methods that can speed up the mini-batch learning method. In this thesis the momentum method and adaptive learning rates are used to produce improved training.

4.3.1. Momentum

Consider the learning process as a particle following the gradient on the error surface. The particle starts off by following the steepest descent, but once it has velocity, it does not go in the steepest way. Its momentum makes it keep going in the previous direction. The momentum method can damp the oscillations effect that happen in high curvature by combining gradients with opposite signs. It also builds up speed in directions with a gentle but consistent gradient:

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \varepsilon \frac{\partial E}{\partial w_{ij}}(t) \tag{4-2}$$

where $\Delta w_{ij}(t-1)$ is the previous momentum, α is a constant, $\frac{\partial E}{\partial w_{ij}}(t)$ is the current gradient and ε is a learning rate.

If the process goes to infinity, the following equation holds:

$$\Delta w_{ij}(\infty) = \frac{1}{1-\alpha}\left(-\varepsilon \frac{\partial E}{\partial w_{ij}}(t)\right) \tag{4-3}$$

Equation (4-3) shows that if the momentum is close to 1, it would be much faster than the simple gradient descent. At the beginning of learning there may be a very large gradient, so it's beneficial to use a small momentum. At the later stage of training the large gradients have disappeared and the weights are stuck in a valley, so the momentum can be gradually increased to 0.9 or even 0.99.

4.3.2. Adaptive learning rate

In a multilayer net, the appropriate learning rates for different parameters can vary widely between weights. The magnitudes of the gradients are often very different for different layers, especially if the initial weights are far from the global minimum. Gradients can get very small in the early layers of very deep nets. The input of each layer often varies widely between layers. Therefore, the learning rate of each parameter in each layer should not behave the same, and different parameters should be trained with different learning rates. It is natural to use a manually set learning rate multiplied by an appropriate local gain empirically determined for each weight.

Here is one example to determine the individual learning rate for different parameters. Start with a local gain of 1 for every weight. Increase the gain if the gradient for that weight does not change sign against the previous one, and scale down a bit if the sign changes. Increases can be additive and decreases can be multiplicative. This ensures that big gains decay

rapidly when oscillations start. It also ensures that in wildly changing gain situations the gain would hover around 1 when increased by δ half of the time and decrease by a factor $1-\delta$ the other half of the time. δ is a positive constant close to 0 (e.g. 0.05). We have

$$\Delta w_{ij} = -\varepsilon g_{ij} \frac{\partial E}{\partial w_{ij}} \tag{4-4}$$

in which ε is a learning rate (manually set learning rate), and g_{ij} is the adaptive learning rate for parameter w_{ij} :

$$\begin{aligned} &if\left(\frac{\partial E}{\partial w_{ij}}(t) \frac{\partial E}{\partial w_{ij}}(t-1) \right) > 0 \\ &then\ g_{ij}(t) = g_{ij}(t-1) + \delta \\ &else\ g_{ij}(t) = g_{ij}(t-1) * (1-\delta) \end{aligned} \tag{4-5}$$

There are some things that should be noticed when using adaptive learning rates. g_{ij} should be prevented to go too far in one direction, and it should be limited in a reasonable range, e.g. [0.1, 10]. It is better to use full batch learning or big mini-batches to ensure that changes in the sign of the gradient are not mainly due to sampling errors in the mini-batch samples. The momentum method can be used in adaptive learning rate too.

Chapter 5. Pretraining of Deep Neural Network

This chapter mainly focuses on the pre-training techniques for DNN. These techniques have a great impact on DNN under many conditions. There are several ways to initialize (or pre-train) DNN, including restricted Boltzmann machine (RBM) which is used in this thesis, and other techniques like deep belief network (DBN) and denoising autoencoder [44].

5.1. Restricted Boltzmann Machine (RBM)

5.1.1. Boltzmann Machine and restricted Boltzmann Machine

A Boltzmann Machine is a kind of energy based network. Energy based networks have a function defining its energy. The output of this kind of network is defined as the configuration of the output units which causes the network to have the lowest energy with given input units.

A Boltzmann Machine contains visible units (input units), hidden units (output units) and symmetric connections between all the units. All units (both visible and hidden) only have two states, 0 or 1. Figure 17 shows a General Boltzmann Machine with 4 visible units (v) and 3 hidden units (h).

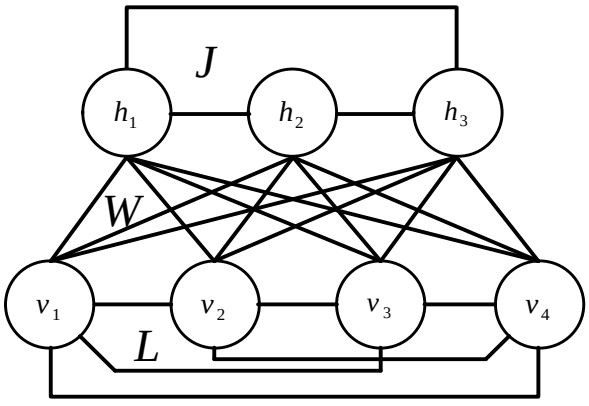


Figure 17 General Boltzmann Machine with 4 visible units and 3 hidden units

The energy of a General Boltzmann Machine with state vector $\mathbf{S} = \{ v_1, v_2, v_3, \ldots, v_n, h_1, h_2, \ldots, h_m \}$ (where *n* is the number of visible units, *m* is the number of hidden units) can be defined as below :

$$E(v,h;\theta) = -vb_v - hb_h - \frac{1}{2}v^T Lv - \frac{1}{2}h^T Jh - \frac{1}{2}v^T Wh \qquad (5\text{-}1)$$

Here, $\theta = \{W, L, J\}$ are the model parameters (or weights, links between the units or neurons), b_v is the bias of visible units, b_h is the bias of hidden units, W represents visible-hidden symmetric interaction terms, and L and J represent the visible-visible and hidden-hidden symmetric interaction terms, respectively.

A Restricted Boltzmann Machine is a simplified kind of Boltzmann Machine with no connection of visible-visible units and hidden-hidden units. Figure 18 shows a Restricted Boltzmann Machine with the same structure as the General Boltzmann Machine showed in Figure 17.

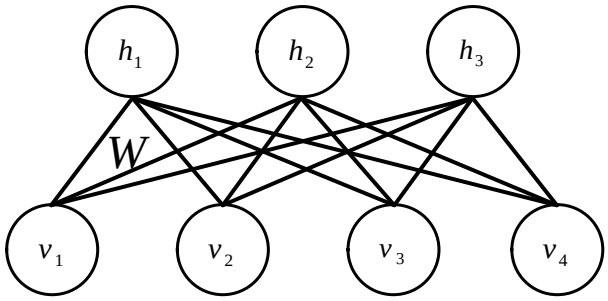


Figure 18 Restricted Boltzmann Machine with 4 visible units and 3 hidden units

In a Restricted Boltzmann Machine a visible unit only has connections to hidden units, and vice versa a hidden unit only has connections to visible units. This special kind of structure provides the advantage that when a visible unit is learning its optimal weights corresponding to a set of hidden units, the learning is independent to other visible units. This advantage also applies to the training of hidden units. Then the whole network can be trained in parallel. With

the great progress made in graphics processing unit (GPU) processors, in other words in parallel computing, this becomes a great advantage.

The energy of a Restricted Boltzmann Machine can be written as below:

$$E(\mathbf{s}) = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij} \tag{5-2}$$

where $\mathbf{s}$ is the state vector as defined before. s_i is a component of $\mathbf{s}$ that can be a visible unit or a hidden unit. If s_j is the unit connected to s_i, then w_{ij} is the weight between s_i and s_j.

5.1.2. Restricted Boltzmann Machine Learning

Let unit i be a unit to update its binary state. The total input z_i for this unit i is the sum of its bias b_i, and the weights from connections to other units:

$$z_i = b_i + \sum_j s_j w_{ij} \tag{5-3}$$

where w_{ij} is the weight of the connection between unit i and unit j.

The probability for this unit to turn on or off is given by a logistic function:

$$prob(s_i = 1) = \frac{1}{1 + e^{-z_i}} \tag{5-4}$$

As discussed before, the visible units only connect to hidden units. For a given state vector $\mathbf{s}$, no matter if the network is updated in any order, the network will eventually reach a stationary distribution (equilibrium). Then for all possible binary state vectors $\mathbf{u}$, the probability of vector $\mathbf{s}$ can be given by the energy:

$$P(\mathbf{s}) = e^{E(\mathbf{s})} \Big/ \sum_{\mathbf{u}} e^{E(\mathbf{u})} \tag{5-5}$$

Given a training set of state vectors (data), the goal of the learning is to find the optimal weights and biases to make the state vectors maximize the product of the probabilities that the

Boltzmann machine assigns to the binary vectors in the training set. By differentiating Eq. 5 and using $\partial E(\mathbf{s}) / \partial w_{ij} = -s_i s_j$, it can be shown that

$$\sum_{\mathbf{s}} \frac{\partial \log P(\mathbf{s})}{\partial w_{ij}} = \left\langle s_i s_j \right\rangle_{data} - \left\langle s_i s_j \right\rangle_{model} \tag{5-6}$$

where $\left\langle s_i s_j \right\rangle_{data}$ is the state value of the data distribution and $\left\langle s_i s_j \right\rangle_{model}$ is the state value when the Boltzmann machine is sampling state vectors from its equilibrium distribution. Then the gradient ascent is surprisingly simple, because the differentiation $\sum_{\mathbf{s}} \frac{\partial \log P(\mathbf{s})}{\partial w_{ij}}$ only depends on the states:

$$\Delta w_{ij} \propto \left\langle s_i s_j \right\rangle_{data} - \left\langle s_i s_j \right\rangle_{model} \tag{5-7}.$$

The update value Δw_{ij} is the product of $\left\langle s_i s_j \right\rangle_{data} - \left\langle s_i s_j \right\rangle_{model}$ and the learning rate. The learning rate can be constant or it can vary during the training steps to satisfy different situations.

To get the Boltzmann equilibrium distribution, we can follow the steps below:

1) Starting with a data vector on visible units, update all of the hidden units in parallel.
2) Update all of the visible units in parallel to get a "reconstruction" of the visible units.
3) Update all of the hidden units again until it is the equilibrium distribution.

This algorithm may take a long time to get to the equilibrium. It can be stopped at step 3) or continue iteratively to update 1) and 2), this is called "contrastive divergence" and has been found to work well in practice [45].

5.1.3. Initialize DNN using RBM

If we stack a few RBMs together we can get a deep layer network, and then use this network as a Neural Network. It becomes a Deep Neural Network, since it has the familiar feed

forward structure and update rule between the layers. Training the stacked RBM is simple, as it is only needed to train the RBM layer by layer.

After training the stacked RBM, we apply/copy the weight vector to the DNN. The RBM weights can be used as the initial weights in a sigmoidal DNN. The conditional probability in the RBM has the same form as in the DNN if the sigmoid nonlinear activation function is used. Then the weights will be closer to the global minima of the DNN and classic backpropagation training is more likely to work well to train the DNN. Note that during the training of the RBM, only the data is used, i.e., the label of the data is not used, it is thus unsupervised training. The DNN is then trained as a general NN using backpropagation, i.e., with supervised training.

Unsupervised pre-training of DNN often improves the performance of DNN training mainly due to three reasons. Firstly, the DNN is highly nonlinear and non-convex, and having the supervised learning start from a good initialization point can greatly affect the final model. Secondly, the unsupervised training criterion used in the pre-training stage is different from the supervised training criterion. Starting from unsupervised pre-trained model thus implicitly regularizes the model. Thirdly, since the pre-training only uses unlabeled data which is much easier to get than labeled data, there is potential for pre-training DNNs with a large quantity of unlabeled data [46].

5.2. Other pre-training method

There are a lot of methods to initialize DNN, like the deep belief network (DBN) and the denoising autoencoder.

A DBN is a generative graphical model, or alternatively a type of deep neural network, composed of multiple layers of hidden units, with connections between the layers but not between units within each layer[47]. The DBN pre-training does not do a great job on networks with only one hidden layer, and it generally works better on networks with two hidden

layers[48][49]. When the number of hidden layers increase, the effectiveness often decrease. As the number of layers increase, the integrated errors increase, and the effectiveness of DBN-pre-training decrease.

An autoencoder [18] is a NN used for learning efficient coding. The aim of an autoencoder is to learn an encoding for a set of data, typically for the purpose of dimensionality reduction[50]. The denoising autoencoder pre-training has similar properties to the DBN pre-training procedure. It is an unsupervised training process and does not require labeled data.

Chapter 6. Input Features for the DNN

There are numerous speech feature and corresponding algorithms that have been used in different speech recognition and/or speech segregation problems. Finding out features that highly correspond to speech intelligibility features, and using these features as the input of a DNN can lead to a sharp increase in the result of DNN training for speech intelligibility enhancement. The MRCG feature set[15] has recently been tested and was found to produce a good result for DNN-based speech intelligibility enhancement. In this chapter, the procedure for extracting the MRCG feature set is presented. Some other possible features are also mentioned in this chapter.

6.1. Multi-Resolution Cochleagram (MRCG) feature

The MRCG feature was originally proposed in [15]. The MRCG feature is a multi-resolution power distribution in the time-frequency representation of an acoustic signal. The cochleagram represents the excitation pattern on the basilar membrane in the inner ear as a function of time. Four cochleagrams at different frequency resolutions are combined to form the MRCG feature, including one high resolution cochleagram and three low resolution cochleagrams.

The cochleagram is calculated in two steps. The input signal is first filtered by a gammatone filter bank:

$$g_{f_c}(t) = t^{N-1}\exp[-2\pi b(f_c)]\cos(2\pi f_c t)u(t) \tag{5-8}$$

where f_c , the center frequencies, are uniformly spaced on the equivalent rectangular bandwidth (ERB) scale, N is the order of the filter, $u(t)$ is the step function, and $b(f_c)$ is the bandwidth related to f_c :

$$b(f_c) = 1.019 * ERB(f_c) = 1.019 * 21.4 * \log(1 + 0.00437 * f_c) \quad (5\text{-}9).$$

$b(f_c)$ increases as f_c increases, which means that the frequency resolution decreases as the frequency increases.

The signal is then divided into frames. A cochleagram is the power of each time frame in each gammatone bank channel.

The MRCG feature can be described as below:

1) Given an input, compute a 64-channel cochleagram (CG1) using 20 ms frames with 10 ms frame shifts, and apply a log operation on the output in each time-frequency (T-F) unit
2) CG2 is similar as CG1, but with 200 ms frames and 10 ms frame shifts
3) CG3 is derived by averaging CG1 across a square window of 11 frequency channels and 11 time frames centered at a given T-F unit. Zero padding is applied at the edges of CG1 when units outside CG1 are needed in the averaging process.
4) CG4 is similar as CG3, but with a 23*23 square window
5) Concatenate CG1-4 to obtain the MRCG feature.

Delta and double-delta features (i.e., computing the differences of feature values at consecutive times, and computing the differences of those differences) are widely used in speech processing to capture temporal dynamics. Delta and double-delta features were thus also used on the MRCG features generated above. Then the final MRCG feature set is obtained. Each time frame becomes an MRCG feature set of dimension 64 by 12.

6.2. Other features

A few of the other popular features for speech intelligibility enhancement are introduced in this section: the Amplitude Modulation Spectrogram (AMS), the Perceptual Linear

Prediction (PLP), the Relative Spectral Transform-PLP (RASTA-PLP), and the Mel-Frequency Cepstral Coefficient (MFCC).

The AMS feature was first used in speech segregation problems in 2009, by Kim, et al [51]. First, the incoming sound signal is processed by a Short-Time Fourier transform (STFT) and the envelope of the resulting spectrogram is taken by squaring the magnitude of the complex values. The frequency scale is then decomposed into a set of critical bands (Bark scale decomposition) and the long-term spectral envelope of each sub-band is analyzed by a second STFT. Thus, the three dimensional (time, acoustic and modulation frequencies) complex AMS coefficients are obtained [52].

The PLP features use three concepts from the psychophysics of hearing to derive an estimate of the auditory spectrum: (1) the critical-band spectral resolution, (2) the equal-loudness curve, and (3) the intensity-loudness power law. The auditory spectrum is then approximated by an autoregressive all-pole model. A 5^{th} order all-pole model is effective in suppressing speaker-dependent details of the auditory spectrum (smoothing). In comparison with conventional linear predictive (LP) analysis, PLP analysis is more consistent with human hearing [53].

RASTA filtering[54] is often coupled with PLP for robust speech recognition. RASTA-filtering was introduced to support PLP preprocessing. It uses bandpass filtering in the log spectral domain. RASTA filtering then removes slow channel variations. It has also been applied to cepstrum feature-based preprocessing, with both log spectral and cepstral domain filtering.

MFCCs are the coefficients that collectively make up the Mel-Frequency Cepstrum (MFC). They are derived from a cepstral representation of the nonlinear acoustic spectrum. MFCCs are derived as follows: 1. Take the Fourier transform of a (typically windowed) signal. 2. Map the powers of the spectrum obtained above onto the mel scale, using triangular

overlapping windows. 3. Take the logs of the powers in each of the mel frequencies. 4. Take the discrete cosine transform of the sequence of mel log powers. 5. The MFCCs are the amplitudes of the resulting transform.

6.3. Procedure for MRCG-DNN speech intelligibility enhancement

With MRCG features as the input and the ideal binary mask (IDBM) as the desired signal, we can form a DNN training setting. The IDBM is a T-F unit mask constructed from the known speech and noise signals and their power ratio (defined in equation (2-23)).

The DNN training process is as below:

1) Use the MRCG features as the input for a stacked RBM. The structure of the stacked RBM is set as four layers: one input layer, two hidden layers, and one output layer. Use contrastive divergence for training. First train the RBM at the bottom (with the input layer), then train the RBM layer on top of it using the output of the previous RBM, and go on until the training of all the RBMs is done. For each RBM training, divide the features into mini batches of 100 features each, and train each mini batch at once. In each mini batch, an average of the gradient is computed and a single update to the weights is performed. Processing all the mini batches and performing the resulting weight update in each mini batch consists of an iteration. Train during 5 iterations for each RBM, with a learning rate set to 1.
2) Set the DNN model to be the same as the stacked RBM, i.e., four layers: one input layer, two hidden layers, and one output layer. Copy the coefficients of the stacked RBM model into the DNN, and use this model as the starting point of the DNN training. In the DNN training, the mini batch size is still 100 features. Train for 100 iterations using the backpropagation. The learning rate is set to 0.5 at the beginning, and then gradually reduced down to 0.1. In order to get a better result and prevent over-fitting,

a validation technique is used. Only 85% of the total data is used as training, the last 15% of the data is used as validation. Choose the best model from the validation data set instead of the training data set. After all the training iterations, save the best model from the validation data as the final model.

3) Use the best model from the validation data (previous step) as the model for test data. Compute the feedforward step of the DNN model, the result is the T-F unit mask estimated by the DNN. Use this mask on the noisy test files T-F units, and then perform synthesis from the T-F units to produce the output processed sound file.

Chapter 7. Simulations

To test how the MRCG-DNN method performs in different situations, we did some tests using the HINT speech database [55], the TIMIT speech database (LDC93S1) [56], and the Aurora noise database. The HINT speech files have a 2 to 3 seconds length for each file, all from the same male speaker. The TIMIT is a corpus of phonemically and lexically transcribed speech of American English speakers of different genders and dialects. Each sentence is about 3 to 5 seconds long. The Aurora noise database contains different situations of noise, including babble, airport, restaurant, and street. In the tests we have used these four kinds of noise.

In this thesis, the MATLAB code is based on two Deep Learning toolboxes, DeepLearnToolBox [57] and DeepNeuralNetwork [58]. The bone structure of the first toolbox, a few active functions of the second toolbox, and some improvements were combined together to produce a new toolbox. The new toolbox was used to perform the simulations of this chapter.

In the following tests, 200 clean speech files are used for training (approx. 5-6 minutes of recordings), and 70 clean speech files are used for testing (approx. 2-3 minutes of recordings). If the training data is too short (i.e., less than 1 min) the result was found to be poor. If the training data is too long it can either cause the computer to run out of memory or take too much processing time. Using 5-6 minutes of recordings for training was found to produce a good result and with a relatively fast processing speed. In [46], the author used 390 sentences, and in [15] the authors used 100 sentence for training. The sampling rate was set to 16000 Hz. The noisy files were produced by randomly selecting a noise segment that has the same length as the clean speech files and adding them together with the desired SNR. The tests were operated using Matlab on a Windows 7 (64-bit) system, with an Intel Core(TM) i5-4310M CPU, and 8 GB memory.

A method to evaluate the estimated mask compared to the IDBM is used here, called the HIT - FA (HIT minus FA) metric, which has been shown to correlate well with human intelligibility [51]. The HIT is the probability of correct detection (the percentage of target-dominant T-F units correctly classified), while the FA is the probability of false alarm (the percentage of noise-dominant units incorrectly classified). Both HIT and FA need to be taken into account when evaluating the performance of the binary mask estimate, and the HIT-FA is a simple difference metric that can be used to quantify the performance of the estimate.

7.1. Tests for simple noises

This test used the HINT database which is a single speaker database. White, purple and pink noises were used here. The model was trained with -5dB, 0 dB and 5dB input SNR, and tests were performed with -5dB SNR. For each noise, the DNN model was trained with speech from the training set with additive noise consisting of only the same noise type. The testing phase was performed with speech from the test set and again additive noise consisting of only the same noise type. Table 4 shows the HIT-FA results and links to the clean test male speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
Noise type	HIT	FA	HIT-FA	noisy	processed
white	84.2%	2.6%	81.6%	link	link
purple	92.3%	3.1%	89.2%	link	link
pink	78.5%	20.2%	58.3%	link	link

Table 4 Results for simple noises

From Table 5 we can see that the DNN model performs very well for white noise, and even better for purple noise. The HIT rate is very high, while the FA rate is very low. The processed sound files are also quite clear. On the other hand, the DNN model produces a much higher FA for pink noise, and the processed file is highly distorted. The reason for this is that

purple noise energy is mainly in high frequency bands, while pink noise is mainly in low frequency bands, and human voice is also mostly distributed in low frequencies. In the purple noise case, the DNN model can easily separate them, it behaves as a kind of high-end low pass filter. In the pink noise condition, the separation is more difficult for the DNN because of the increased frequency overlap between speech and noise.

7.2. Same speaker for training and testing, same noise type for training and testing

Here we used the HINT database which has a single speaker, with different sentences for training and testing. We used four kinds of realistic noise: airport, babble, restaurant, and street. We trained the DNN model with -5dB, 0 dB and 5dB input SNR, and then performed the testing with -5dB SNR. For each type of noise we trained the model and then used the model to test different sentences corrupted by the same kind of noise as the one used for training (but different noise segments). Table 5 shows the HIT-FA results and links to the clean test male speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
Noise type	HIT	FA	HIT-FA	noisy	processed
babble	76.2%	7.3%	68.9%	link	link
airport	80.3%	11.0%	69.3%	link	link
restaurant	77.5%	9.9%	67.6%	link	link
street	78.2%	7.5%	70.7%	link	link

Table 5 Results for same speaker for training and testing, same noise type for training and testing

From Table 5 we can see that the HIT rate of the estimated mask is relatively high and the FA rate is low, regardless of the noise type. The resulting HIT-FA rate are fairly good and are comparable with previous results from the literature[46]. However, when listening to the processed output files it is debatable if the intelligibility is really improved compared to the original noisy files. In some cases (e.g. babble noise) the intelligibility of the processed files

appears to be better than for other cases (e.g. street noise), so this indicates that the performance is noise environment-specific. It should be noted that the noises used for testing were challenging noises, but they are more likely to correspond to real noise conditions in practice.

7.3. Different speakers for training and testing, same noise type for training and testing

In this section, we used the TIMIT dataset with several different speakers for training and another speaker was used for testing (but same female gender as the ones used in training). Four kinds of noise were used: airport, babble, restaurant, and street. We trained the DNN model with -5dB, 0 dB and 5dB SNR, and did testing with -5dB SNR. For each type of noise we trained the model and performed the testing using the same type of noise (but different noise segments). Table 6 shows the HIT-FA results and links to the clean test female speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
Noise type	HIT	FA	HIT-FA	noisy	processed
airport	83.8%	8.2%	75.6%	link	link
babble	80.6%	6.9%	73.7%	link	link
restaurant	82.2%	13.2%	69.0%	link	link
street	84.4%	6.0%	78.4%	link	link

Table 6 Results for different speakers for training and testing, same noise type for training and testing

From Table 6 we can see that the HIT rate of the estimated mask is relatively high and the FA rate is low, regardless of the noise type. This indicates that the performance of the MRCG-DNN method can be robust to different speakers, i.e., when the speakers used for training differ from the speakers used in testing. Although it is possible to train simultaneously for both male and female speech, our experience has been that better results are obtained when the same gender is used for training and testing. In terms of intelligibility, as in the results of Table 5, it

is debatable if the intelligibility is improved in the processed sound files of Table 6, and the performance seems to be better for some noise types (e.g. airport and street).

7.4. Training with 3 types of noise and testing with a fourth type of noise

For this test, we have used either the HINT dataset (single speaker) or the TIMIT dataset (different speakers), as well as airport, babble, restaurant, and street noise. We trained the DNN model with airport, babble and restaurant noise, then we tested the model with street noise. For the HINT database the training and testing sentences were different but from the same speaker. For the TIMIT database, training and testing were made with different speakers (of the same gender) and different sentences. Table 7 shows the HIT-FA results and links to the clean test speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
	HIT	FA	HIT-FA	noisy	processed
HINT database	76.2%	11.4%	64.8%	link	link
Clean test speech					
	HIT	FA	HIT-FA	noisy	processed
TIMIT database	84.6%	18.9%	65.7%	link	link

Table 7 Results for training with 3 types of noise and testing with a fourth type of noise

From Table 7, we can see that the HIT rate of the estimated mask is relatively high and the FA rate is low. This indicates that in principle the performance of the MRCG-DNN method is robust to conditions where we have both different speakers and different noise types between training and testing, as long as training is done with appropriate data (i.e., the training speech should have some similarity with the test speech, the training noise should have some similarity with the test noise). But as observed in the previous results of Table 5 and Table 6, in the results of Table 7 it is highly debatable if the intelligibility is improved.

7.5. Training with different levels of noise, testing with the same noise type at a specific level

Here we used the HINT dataset (same speaker for training and testing, but different sentences), and babble noise for training and testing. We trained the DNN model with -5dB, 0 dB and 5dB SNR, then tested with -5dB SNR. We trained the DNN model with -7dB, -5 dB and 0dB SNR, then tested with -5dB SNR. Finally, we trained the DNN model with -10dB, -7 dB and -5dB SNR, and tested with -5dB SNR. Table 8 shows the HIT-FA results and links to the clean test speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
SNR	HIT	FA	HIT-FA	noisy	processed
-5dB, 0dB, 5dB	76.2%	7.3%	68.9%	link	link
-7dB,-5dB, 0dB	74.8%	6.6%	68.1%	link	link
-10dB,-7dB,-5dB	70.3%	5.3%	64.9%	link	link

Table 8 Results for training with different levels of noise, testing with the same noise type at a specific level

From Table 8 we can see that the HIT rate as well as the FA and HIT-FA rates of the estimated mask all slightly drop from the top row to the bottom row. This indicates that training with data having the same and higher SNR than the testing data may lead to a slightly higher HIT-FA rate. Informally, the intelligibility in the sound files also seems to follow that trend.

7.6. Training with different local SNR criterion (i.e., thresholds used for binary mask decision)

For this test, we used the HINT dataset (same speaker for training and testing, but different sentences), and babble noise for training and testing. We trained the DNN model with -5dB, 0 dB and 5dB SNR, and tested with -5dB SNR. We did this procedure 3 times, and the difference between the 3 cases is that different local SNR criterion values were used (thresholds used for

the binary mask decision): 0dB, -5dB, and -10dB. Table 9 shows the HIT-FA results and links to the clean test speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
Local Criterion	HIT	FA	HIT-FA	noisy	processed
0dB	74.3%	8.8%	65.5%	link	link
-5dB	76.2%	7.3%	68.9%	link	link
-10dB	79.9%	11.7%	68.3%	link	link

Table 9 Results of training with different local SNR criterion (thresholds)

From Table 9 we can see that the HIT rate of the estimated mask increases as the local SNR criterion (threshold) decreases, but the HIT-FA rate doesn't change too much. Listening to the resulting processed files, we note that the intelligibility improves from the top row to the bottom row. A higher HIT rate means that a higher percentage of target-dominant T-F bins are kept, so based on these results we adopted a dynamic adjustment method during training in the software program, to make sure that the HIT rate is at least 75%. If the condition is not fulfilled then we dynamically lower the local SNR criterion used for training. But overall, as in previous tables, it is debatable if the processed files provide some intelligibility improvement over the original noisy files.

7.7. Reverberation testing

In this part we tested how DNN speech intelligibility enhancement behaves in high reverberation conditions, compared to conditions without reverberation. The HINT database is used, with the same speaker for training and testing, but different sentences. The speech training set sentences and test set sentences were passed through a reverberation filter (T60 approx. 900ms), and then they were used as the new speech training set and speech test set. White noise and Speech-Shaped-Noise (SSN) were used for this test. The model was trained with -5dB, 0 dB and 5dB input SNR, and testing was performed with -5dB SNR. Table 10

shows the HIT-FA results and provides links to the clean test speech sound file, the noisy sound files and the processed sound files.

Clean test speech					
Noise type	HIT	FA	HIT-FA	noisy	processed
white	80.1%	15.0%	65.1%	link	link
SSN	82.7%	27.4%	55.3%	link	link

Table 10 Results for reverberation test

From Table 10, we see that the FA rates are higher than for most previous results, which seems to indicate that DNN speech intelligibility enhancement doesn't perform well for reverberant signals. However, listening to the noisy files we also see that the noisy files were quite challenging (low intelligibility), despite being at the same -5 dB SNR as previous results. So this can also be an explanation for the higher FA values in Table 10. But in any case, there is no doubt that the processed files are quite unintelligible.

7.8. MRCG-DNN compared with other approaches

In this section we use two other types of speech intelligibility enhancement approaches and compare them with the MRCG-DNN approach: an 85-D features DNN proposed by Healy, Eric W., et al [24] and an AMS-GMM proposed by Kim, et al [46]. We use our own implementation of an 85-D features DNN and the original code from Kim's AMS-GMM. The database and noise conditions were the same as the ones originally used in the 85-D features DNN and the AMS-GMM. The IEEE speech database is also used as clean speech data in this section [59].

For the DNN with 85-D features, the input data is passed through a 64 band gammatone filter. Each subband is divided to 20 ms frames with 10 ms overlap. Each T-F unit extracts 85-D features: 15 AMS features, 13 RASTA-PLP features, 31 MFCC features, and 13 delta features for the RASTA-PLP features. This method uses a DNN model for each subband, for

a total of 64 DNN model. The MRCG-DNN on the other hand only uses one DNN model for all subbands, so the MRCG-DNN training take much less time for training than the 85-D-DNN.

The results for the 85-D-DNN and the MRCG-DNN are compared in Table 11. They both use the same DNN structure, were trained for 100 iterations. "n6 noise" is a babble noise produced by adding several TIMIT sentences together. We can see from Table 11 that the HIT-FA for the MRCG-DNN is always significantly higher than for the 85-D-DNN.

		85-D-DNN			MRCG-DNN		
database	noise	HIT	FA	HIT-FA	HIT	FA	HIT-FA
HINT	n6	62.9%	7.4%	55.5%	72.3%	6.4%	65.9%
IEEE	factory	46.2%	3.4%	42.8%	66.5%	3.8%	62.7%

Table 11 Results for 85-D-DNN and MRCG-DNN.

The other method used for comparison is a Gaussian Mixture Model (GMM) using AMS features. More specifically, in this approach 4 sub-GMMs are used in order to make the decision to set the TF binary mask to 1 or 0. The results for the AMS-GMM and the MRCG-DNN are compared in Table 12. The HIT-FA of the MRCG-DNN is slightly less (worse) than for the AMS-GMM method, but with a much lower FA (better). Overall it can be said that the results of these two methods are fairly comparable for the considered setup. However, we also found that the AMS-GMM method fails to find a clustering solution in some conditions (e.g., IEEE database speech with factory noise), while the MRCG-DNN method always converges to a solution, so it was found to be much more robust.

		AMS-GMM			MRCG-DNN		
database	noise	HIT	FA	HIT-FA	HIT	FA	HIT-FA
IEEE	speech shape noise	93.4%	12.3%	81.12%	80.5%	3%	77.4%

Table 12 Results for AMS-GMM and MRCG-DNN.

In conclusion, the MRCG-DNN has a better HIT-FA than the DNN with other features (85-D feature), and it is more robust than the AMS-GMM method, with similar performance. In addtion, the MRCG-DNN method is faster than the 85-D-DNN and AMS-GMM methods. The MRCG-DNN takes about half an hour of training time, whilethe 85-D-DNN and AMS-GMM both need more than three hours of training time. Therefore, compared to other available methods, the MRCG-DNN was confirmed as a good choice for attempting to improve speech intelligibility (despite the debatable improvement in intelligibility of the processed sound files in previous tables).

Chapter 8. Conclusion

Single-channel speech intelligibility improvement is more difficult than multi-channel speech intelligibility improvement, because we don't know how and to what extend to adjust the spectrum, and there is no spatial information available. Traditional SNR-based speech enhancement algorithms (e.g. Wiener filter, spectral subtraction, log-MMSE STSA) can't improve speech intelligibility, and a few tentative reasons to explain this were discussed. On the other hand, single channel data-dependent and machine learning based algorithms have recently been proposed to improve speech intelligibility.

In this thesis, we implemented and tested the Deep Neural Network approach for speech intelligibility improvement. A series of experiments have been presented in the thesis. Based on objective measures (HIT-FA rate), the performance of the implemented MRCG-DNN approach was found to be competitive compared to previous results appearing in the literature. The HIT-FA results that were obtained were reasonably good for different settings, such as for the single speaker setting (same speaker used for training and testing, for different noise types), the case where the training is done with multiple speakers and the testing is done with another speaker (for different noise types), and the case where the training is done with different types of noise and the testing is done with another noise type. However, in most of the processed files, it is debatable (and in some cases highy debatable) if the processing has introduced an intelligibility improvement or not.

Using the DNN method with "easier" noises (white noise, purple noise) lead to a better performance because of the stationarity of the noise and the reduced overlap with the speech content. Training with more data (clean speech data, noise data, or both) could be another

option to attempt to improve the performance, although that option may not always be feasible in real-life scenarios.

References

[1] https://en.wikipedia.org/wiki/Intelligibility_(communication)

[2] Niederjohn, R. J., and Grotelueschen, J. H. "The enhancement of speech intelligibility in high noise levels by high-pass filtering followed by rapid amplitude compression." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 24.4 (1976): 277-282.

[3] Skowronski, M. D., and Harris, J. G. "Applied principles of clear and Lombard speech for automated intelligibility enhancement in noisy environments." *Speech Communication* 48.5 (2006): 549-558.

[4] Yoo, S. D., Boston, J. R., El-Jaroudi, A., Li, C. C., Durrant, J. D., Kovacyk, K., and Shaiman, S. "Speech signal modification to increase intelligibility in noisy environments." *The Journal of the Acoustical Society of America* 122.2 (2007): 1138-1149.

[5] McLoughlin, I. V., & Chance, R. J. " LSP-based speech modification for intelligibility enhancement" . *13th International Conference on DSP*. IEEE. (1997) : 591-594.

[6] Huang, D. Y., Rahardja, S., and Ong, E. P. "Lombard effect mimicking." *SSW7 The Seventh ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*. (2010) : 258-263.

[7] Langner, B., and Black, A. W. "Improving the Understandability of Speech Synthesis by Modeling Speech in Noise." *ICASSP* (2005) : 265-268.

[8] Loizou, P. C. *Speech enhancement: theory and practice*. CRC press, 2013.

[9] Tang, Y., and Cooke, M. "Energy reallocation strategies for speech enhancement in known noise conditions." *Interspeech*. (2010) : 1636-1639.

[10] Tang, Y., and Cooke, M. "Optimised spectral weightings for noise-dependent speech intelligibility enhancement." *Interspeech*. (2012).

[11] Valentini-Botinhao, C., Maia, R., Yamagishi, J., King, S., and Zen, H. "Cepstral analysis based on the Glimpse proportion measure for improving the intelligibility of HMM-based synthetic speech in noise." *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, (2012) : 3997-4000.

[12] Taal, C. H., Hendriks, R. C., and Heusdens, R. "A speech preprocessing strategy for intelligibility improvement in noise based on a perceptual distortion measure." *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, (2012): 4061-4064.

[13] Petkov, P. N., and Kleijn, W. B. "Spectral dynamics recovery for enhanced speech intelligibility in noise." *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 23.2 (2015): 327-338.

[14] Kim, G., and Loizou, P. C. "Improving speech intelligibility in noise using environment-optimized algorithms." *Audio, Speech, and Language Processing, IEEE Transactions on* 18.8 (2010): 2080-2090.

[15] Chen, J., Wang, Y., and Wang, D. "A feature study for classification-based speech separation at very low signal-to-noise ratio."*Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014 : 7039-7043.

[16] Dempster, A., Laird, N., and Rubin, D. "Maximum likelihood from incomplete data via the EM algorithm. " J*ournal of the Royal Statistical Society*, Series B(1977), 39(1):1–38.

[17] Hinton, G. E., and Salakhutdinov, R. R. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507.

[18] Bengio, Y. "Learning deep architectures for AI. " *Foundations and trends in Machine Learning* 2.1 (2009): 1-127.

[19] Hinton, G. E., Osindero, S., and Teh, Y. W. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.

[20] Arel, I., Rose, D. C., and Karnowski, T. P. "Deep machine learning-a new frontier in artificial intelligence research." *Computational Intelligence Magazine, IEEE* 5.4 (2010): 13-18.

[21] Deng, L. "An overview of deep-structured learning for information processing." *Proceedings of Asian-Pacific Signal & Information Processing Annual Summit and Conference (APSIPA-ASC)*. (2011).

[22] Yu, D., and Deng, L. "Deep learning and its applications to signal and information processing." *Signal Processing Magazine, IEEE* 28.1 (2011): 145-154.

[23] Deng, L., and Yu, D. "Deep learning: methods and applications." *Foundations and Trends in Signal Processing* 7.3–4 (2014): 197-387.

[24] Healy, E. W., Yoho, S. E., Wang, Y., and Wang, D. "An algorithm to improve speech recognition in noise for hearing-impaired listeners." *The Journal of the Acoustical Society of America* 134.4 (2013): 3029-3038.

[25] Shannon, R. V., Zeng, F. G., Kamath, V., Wygonski, J., and Ekelid, M. "Speech recognition with primarily temporal cues." *Science* 270.5234 (1995): 303-304.

[26] James, J. H., Chen, B., and Garrison, L. "Implementing VoIP: a voice transmission performance progress report." *Communications Magazine, IEEE* 42.7 (2004): 36-41.

[27] Loizou, P. C., and Kim, G. "Reasons why current speech-enhancement algorithms do not improve speech intelligibility and suggested solutions." *Audio, Speech, and Language Processing, IEEE Transactions on* 19.1 (2011): 47-56.

[28] Miller, G. A., Heise, G. A., and Lichten, W. "The intelligibility of speech as a function of the context of the test materials." *Journal of experimental psychology* 41.5 (1951): 329.

[29] ANSI, ANSI. "S3. 5-1997. *Methods for Calculation of the Speech Intelligibility Index, American National Standards Institute, New York* (1997).

[30] Houtgast, T., and Steeneken, H. J. "The modulation transfer function in room acoustics as a predictor of speech intelligibility."*Acta Acustica united with Acustica* 28.1 (1973): 66-73.

[31] Houtgast, T., and Steeneken, H. J. "A review of the MTF concept in room acoustics and its use for estimating speech intelligibility in auditoria." *The Journal of the Acoustical Society of America* 77.3 (1985): 1069-1077.

[32] Kates, J. M., and Arehart, K. H. "Coherence and the speech intelligibility index."*The Journal of the Acoustical Society of America*117.4 (2005): 2224-2237.

[33] https://en.wikipedia.org/wiki/Artificial_neural_network

[34] https://en.wikipedia.org/wiki/Deep_learning

[35] Rosenblatt, F. "The Perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

[36] LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. "Efficient backpropagation" *Neural networks: Tricks of the trade.* Springer Berlin Heidelberg, 2012. 9-48.

[37] Han, J., & Moraga, C. "The influence of the sigmoid function parameters on the speed of backpropagation learning." *From Natural to Artificial Neural Computation.* Springer Berlin Heidelberg, 1995. 195-201.

[38] Haykin, S.. "Neural Networks - A Comprehensive Foundation, 2nd ed. " *Prentice-Hall, Englewood Cliffs*, 1998.

[39] Bishop, C. M. "Pattern Recognition." *Machine Learning* (2006) :128.

[40] Williams, R. J. , and Hinton, G. E. "Learning representations by back-propagating errors." *Nature* (1986): 323-533.

[41] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. *Learning internal representations by error propagation*. No. ICS-8506. California Univ. San Diego La Jolla Inst. For Cognitive Science, 1985.

[42] Srivastava, N. *Improving neural networks with dropout*. Diss. University of Toronto, 2013.

[43] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[44] Yu, Dong, and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach.* London: Springer, 2015. ISBN: 978-1-4471-5778-6 Print.

[45] Carreira-Perpinan, M. A., and Hinton, G. E. "On contrastive divergence learning." *Proceedings of the tenth international workshop on artificial intelligence and statistics*. NP: Society for Artificial Intelligence and Statistics, (2005) : 33-40.

[46] Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., and Bengio, S. "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research* 11 (2010): 625-660.

[47] https://en.wikipedia.org/wiki/Deep_belief_network

[48] Seide, F., Li, G., Chen, X., and Yu, D. "Feature engineering in context-dependent deep neural networks for conversational speech transcription." *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, (2011) : 24-29.

[49] Seide, F., Li, G., and Yu, D. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks."*Interspeech*. (2011) : 437-440.

[50] https://en.wikipedia.org/wiki/Autoencoder

[51] Kim, G., Lu, Y., Hu, Y., and Loizou, P. C. "An algorithm that improves speech intelligibility in noise for normal-hearing listeners." *The Journal of the Acoustical Society of America* 126.3 (2009): 1486-1494.

[52] Moritz, N., Anemüller, J., and Kollmeier, B. "Amplitude modulation spectrogram based features for robust speech recognition in noisy and reverberant environments." *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, (2011) : 5492-5495.

[53] Hermansky, H. "Perceptual linear predictive (PLP) analysis of speech." *the Journal of the Acoustical Society of America* 87.4 (1990): 1738-1752.

[54] Hermansky, H., and Morgan, N. "RASTA processing of speech." *Speech and Audio Processing, IEEE Transactions on* 2.4 (1994): 578-589.

[55] Nilsson, M., Soli, S. D., and Sullivan, J. A. "Development of the Hearing in Noise Test for the measurement of speech reception thresholds in quiet and in noise." *The Journal of the Acoustical Society of America* 95.2 (1994): 1085-1099.

[56] https://catalog.ldc.upenn.edu/LDC93S1

[57] https://github.com/rasmusbergpalm/DeepLearnToolbox

[58] http://www.mathworks.com/matlabcentral/fileexchange/42853-deep-neural-network

[59] Rothauser, E. H., Chapman, W. D., Guttman, N., Nordby, K. S., Silbiger, H. R., Urbanek, G. E., and Weinstock, M. "IEEE recommended practice for speech quality measurements." *IEEE Trans. Audio Electroacoust* 17.3 (1969): 225-246.